Englisch · Szczepanowski

Das große Floppy-Buch

Disketten-Programmierung mit COMMODORE Computern für Anfänger, Fortgeschrittene und Profis

EIN DATA BECKER BUCH

Englisch · Szczepanowski

Das große Floppy-Buch

Disketten-Programmierung mit COMMODORE Computern für Anfänger, Fortgeschrittene und Profis

EIN DATA BECKER BUCH

I S B N 3-89011-005-3

2. überarbeitete Auflage

Copyright (C) 1984 DATA BECKER GmbH
Merowingerstr. 30
4000 Düsseldorf

Alle Rechte vorbehalten. Kein Teil dieses Buches darf in irgendeiner Form (Druck, Fotokopie oder einem anderen Verfahren) ohne schriftliche Genehmigung der DATA BECKER GmbH reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Wichtiger Hinweis!

Die in diesem Buch wiedergegebenen Schaltungen, Verfahren und Programme werden ohne Rücksicht auf die Patentlage mitgeteilt. Sie sind ausschließlich für Amateur- und Lehrzwecke bestimmt und dürfen nicht gewerblich genutzt werden.

Alle Schaltungen, technische Angaben und Programme in diesem Buch wurden von den Autoren mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. DATA BECKER sieht sich deshalb gezwungen, darauf hinzuweisen, daß weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernommen werden kann. Für die Mitteilung eventueller Fehler sind die Autoren jederzeit dankbar.

V O R W O R T

Mit der Floppy VC 1541 steht dem COMMODORE Anwender für erstaunlich wenig Geld ein sehr leistungsfähiges externes Speichermedium zur Verfügung. Um die vielseitigen Möglichkeiten der VC 1541 ausschöpfen zu können, bedarf es aber entsprechender Informationen und Anregungen. Lothar Englisch und Norbert Szczepanowski haben in monatelanger Kleinarbeit alle Geheimnisse der VC 1541 für Sie ergründet.

Das große Floppy Buch reicht von der einfachen Programmbeschreibung über den anspruchsvollen Direktzugriff bis hin zur Overlaytechnik. Anfänger werden die zahlreichen Beispielprogramme begrüßen, mit denen der Text anschaulich illustriert wird. Maschinenprogrammierer werden insbesondere das ausführlich dokumentierte Listing des Diskettenbetriebssystems (DOS) schätzen und die exakten technischen Beschreibungen. Ein echtes Schnäppchen wird "Das große Floppy Buch" alleine schon durch die große Anzahl komplett lauffertiger Programme, die nur noch eingetippt werden müssen. Neben BASIC-Erweiterungen, hilfreichen Dienstprogrammen und nützlicher Routinen wie z.B. Spooling zählen hierzu vor allem eine leistungsfähige Adreßverwaltung, eine komplette Haushaltsbuchführung und ein komfortabler Disk-Monitor zur Manipulation einzelner Sektoren.

Viel Spaß bei der Lektüre des großen Floppy Buches und bei der Arbeit mit Ihrer Floppy VC 1541.

Dr. Achim Becker

I N H A L T S V E R Z E I C H N I S

Kapitel 1:
Einführung in die Programmierung der VC 1541

1.1 Der erste Kontakt mit der VC 1541

1.1.1 Das Disketten-Betriebssystem 1
1.1.2 Die Test/Demo-Diskette 2
1.1.3 Vorbehandlung neuer Disketten 2
1.1.4 Einige Daten der VC 1541-Diskette 3

1.2 Das Speichern von Programmen auf Diskette

1.2.1 SAVE - Speichern von BASIC-Programmen 4
1.2.2 LOAD - Laden von BASIC-Programmen 5
1.2.3 VERIFY - Überprüfen von gespeicherten Programmen 5
1.2.4 Überschreiben von Programmen 5
1.2.5 Laden von Maschinenprogrammen 6
1.2.6 Speichern von Machinenprogrammen 8

1.3 Die Floppy-Systembefehle

1.3.1 Die Befehlsübermittlung zur Floppy-Station 10
1.3.2 NEW - Formatieren von Disketten 11
1.3.3 Auslesen des Fehlerkanals 12
1.3.4 Laden des Directories 13
1.3.5 SCRATCH - Löschen von Files 15
1.3.6 RENAME - Umbenennen von Files 15
1.3.7 COPY - Kopieren von Files 16
1.3.8 INITIALISE - Initialisieren der Diskette 17
1.3.9 VALIDATE - "Aufräumen" der Diskette 17
1.3.10 Der "Joker" 18

1.4 Sequentielle Datenspeicherung

1.4.1 Das Prinzip 21
1.4.2 OPEN - Eröffnen einer sequentiellen Datei 23
1.4.3 PRINT/INPUT - Datenübertragung Floppy/Rechner 25
1.4.4 Anhängen von Datensätzen 28
1.4.5 CLOSE - Schließen einer sequentiellen Datei 29
1.4.6 "Umleiten" der Bildschirmausgabe 30
1.4.7 Sequentielle Datei als Tabelle im Rechner 31
1.4.8 Suchen in der Tabelle 34
1.4.9 Einfaches Sortieren der Tabelle 37
1.4.10 ADRESSENVERWALTUNG mit sequentieller Daten- 40
speicherung
1.4.11 Anwendungsgebiete der sequentiellen Datenspeicherung .. 47

1.5 Relative Datenspeicherung

1.5.1 Das Prinzip 48
1.5.2 Der Vorteil gegenüber sequentieller Speicherung 49
1.5.3 Das Eröffnen einer relativen Datei 49

1.5.4 Vorbereitung der Daten zur relativen Speicherung 52
1.5.5 Datenübertragung Floppy/Rechner 54
1.5.6 Schließen der relativen Datei 57
1.5.7 Suchen eines Records nach der binären Methode 57
1.5.8 Suchen eines Records über seperate Index-Dateien 61
1.5.9 Ändern eines Records 63
1.5.10 Ergänzen einer relativen Datei 64
1.5.11 Haushaltsbuch mit relativer Datenspeicherung 65

1.6 Die Fehlermeldungen der Floppy und ihre Ursachen 74
1.7 Übersicht aller Befehle mit Vergleich 79
BASIC 2.0 - BASIC 4.0

Kapitel 2:
Programmierung für Fortgeschrittene

2.1 Der Direktzugriff auf jeden Block der Diskette 85

2.2 Die Direktzugriffsbefehle

2.2.1 Der Block-Read-Befehl 88
2.2.2 Der Block-Pointer-Befehl 89
2.2.3 Der Block-Write-Befehl 90
2.2.4 Der Block-Allocate-Befehl 91
2.2.5 Der Block-Free-Befehl 92
2.2.6 Der Block-Execute-Befehl 93
2.3 Anwendungen des Direktzugriffs
2.4 Der Zugriff auf das DOS - Die Memory-Befehle

2.4.1 Der Memory-Read-Befehl 96
2.4.2 Der Memory-Write-Befehl 97
2.4.3 Der Memory-Execute-Befehl 98
2.4.4 Die User-Befehle 99

Kapitel 3:
Technik der Floppy und der Diskette

3.1 Der Aufbau der VC 1541

3.1.1 Blockschaltbild der Floppy 101
3.1.2 Memory-Map des DOS - ROM, RAM, I/O 102

3.2 Die Arbeitsweise des DOS - ein Überblick 106

3.3 Der Aufbau der Diskette

3.3.1 Die BAM der VC-1541 108
3.3.2 Das Directory 109
3.3.3 Das Format der Directorys 111

3.4 Die Organisation der relativen Dateien 117
3.5 DOS-Listing der VC 1541 122

Kapitel 4:
Programme und Tips zur Benutzung der VC 1541

4.1 Dienstprogramme

4.1.1 Anzeige aller Fileparameter 270
4.1.2 Scratch-Schutz von Files - Fileprotect 275
4.1.3 Backup-Programm - Kopieren von Disketten 281
4.1.4 Kopieren einzelner Files auf eine andere Diskette 283
4.1.5 Einlesen des Directorys innerhalb von Programmen 285
4.1.6 Suchen von "versteckten" Files 286

4.2 Die Dienstprogramme der Test/Demo-Diskette

4.2.1 DOS 5.1 288
4.2.2 COPY/ALL 289
4.2.3 DISK ADDR CHANGE 289
4.2.4 DIR 290
4.2.5 VIEW BAM 290
4.2.6 CHECK DISK 290
4.2.7 DISPLAY T&S 291
4.2.8 PERFORMANCE TEST 291

4.3 BASIC-Erweiterungen und Programme zur komfortablen Benutzung der VC 1541

4.3.1 Eingabe beliebig langer Strings von Diskette 292
4.3.2 Komfortables Aufbereiten von Datensätzen 296
4.3.3 Spooling - Direktes Drucken von Diskette 302

4.4 Overlaytechnik und Nachladen von Maschinenprogrammen 305
4.5 Merge - Aneinanderhängen von BASIC-Programmen 308
4.6 Disk-Monitor für Commodore 64 und VC 20 310

Kapitel 5:
Die großen CBM-Floppies

5.1 IEC-Bus und serieller Bus 323
5.2 Gemeinsamkeiten und Unterschiede gegenüber der VC 1541 325

Kapitel 1: Einführung in die Programmierung der VC-1541

1.1 Der erste Kontakt mit der Floppy-Disk

Da steht sie nun, Ihre neue Diskettenstation. Schnell, leistungsfähig und für Sie zunächst ein Buch mit sieben Siegeln. Aber keine Angst. Schrittweise werden wir Sie in die Kunst der Diskettenprogrammierung einführen. Der erste Teil des Buches gibt dem Anfänger einen intensiven Überblick über den Umgang mit der Floppy VC-1541 und über Ihre Programmierung. Jedem Befehl folgt mindestens ein Beispiel, an dem seine Funktion erprobt und erkannt werden kann. Sie werden überrascht sein, wie einfach doch die Handhabung Ihrer Diskettenstation sein kann, wenn man nur im Besitz von guter, begleitender Literatur ist.
Der Anfänger, der das Laufwerk zunächst hauptsächlich zur Speicherung von Programmen nutzen wird und vielleicht beim Kauf auch kein anderes Einsatzgebiet in Erwägung gezogen hat, wird in diesem Buch mit den vielen anderen Möglichkeiten der Diskettenprogrammierung vertraut gemacht. Ein Programmierer, der vorher die Datenspeicherung auf Kassette organisierte wird die wesentlichen Vorteile der Diskette erkennen und einzusetzen lernen.
Auch erfahrene Programmierer sollten sich nicht scheuen, das erste Kapitel intensiv zu erarbeiten, denn sie werden sicherlich Ihre Kenntnisse erweitern. Dies betrifft besonders die relative Dateiverwaltung.

1.1.1 Das Disketten-Betriebssystem

Die Diskettenstation besitzt neben der Laufwerksmechanik und der Elektronik wie der VC-20 und der Commodore 64 ein eigenes Betriebssystem zur Steuerung der internen Vorgänge und zum Ausführen der vom Rechner übersandten Befehle. Dieses DOS (Disk Operating System) genannte Betriebssystem ist auf dem VC-20 und Commodore 64 abgestimmt und trägt die Bezeichnung CBM DOS V2.6 1541. Diese Version V2.6 enthält auch einige zusätzliche Möglichkeiten, die sich mit dem VC-20 und dem Commodore 64 nicht ohne weiters nutzen lassen.

Der Commodore 64 und der VC-20 enthalten das BASIC CBM 2.0. Die VC-1541 hingegen versteht die erweiterten Diskettenbefehle des BASIC 4.0, die sich mit dem BASIC 2.0 simulieren lassen.

Am Ende des Kapitels folgt eine Auflistung sämtlicher Befehle des BASIC 2.0 mit entsprechenden Befehlen des komfortableren BASIC 4.0, wie es bei den größeren CBM-Rechnern (4000-er, 8000-er und der neuen 600-er und 700-er Serie) integriert ist.

Das CBM BASIC 4.0 ist auch auf dem VC 20 und dem COMMODORE 64 einsetzbar. Es ist in folgenden Produkten, die in unserem VC INFO ausführlich beschrieben sind, intergriert:

VC 20: DATA BECKER IEC-Bus mit DISC BASIC
CBM 64: DATA BECKER IEC-Bus mit DISC BASIC
SUPERTWIN
MASTER 64

1.1.2 Die beigelegte Test/Demo-Diskette

Sicher wußten Sie mit dieser Diskette erst gar nichts anzufangen. Das soll Sie aber nicht irritieren. Neben Programmbeispielen enthält diese Diskette Dienstprogramme, die Sie ohne den entsprechenden Kenntnissen des Disketten-Betriebssystems nicht sinnvoll bedienen können. Legen Sie diese Diskette erst einmal beiseite. Im Laufe dieses und des folgenden Kapitels erlernen Sie alles, was das DOS zu bieten hat und werden bald in der Lage sein, selbst ähnliche Dienstprogramme ohne großartige Anstrengung zu schreiben.
Die Test/Demo-Diskette wird später noch ausführlich beschrieben.

1.1.3 Die Vorbehandlung neuer Disketten zum Einsatz

Ladenübliche, "rohe" Disketten müssen erst vorbehandelt werden, bevor man sie zur Datenspeicherung verwenden kann. Diesen Vorgang nennt man in der Fachsprache "Formatieren". Was bedeutet nun "Formatieren"? Jedes typenungleiche Laufwerk hat seine Besonderheiten. So ist die Diskette z.B. in Spuren aufgeteilt, deren Anzahl bei vielen Laufwerken unterschiedlich ist. Außerdem ist jede Spur in Sektoren gegliedert, deren Anzahl ebenfalls zwischen verschiedenen Laufwerken variieren kann. Jeder dieser Sektoren erhält während dem Formatieren seine "Adresse", an der das DOS ihn identifiziert. Diese Adresse besteht aus der fortlaufenden Nummer von Spur und Sektor. Weiterhin wird jeder Sektor mit einem Kode belegt, an dem das DOS erkennt, ob die Diskette auch auf diesem Laufwerktyp formatiert wurde. Dieses Formatkennzeichen besteht aus zwei Zeichen, und enthält bei der VC 1541 "2A". Der Rest des Sektors ("Block" genannt) kann maximal 256 Zeichen aufnehmen.
Die letzte Aufgabe des "Formatierens" ist, das Directory (Inhaltsverzeichnis) der Diskette anzulegen. In der Directory sind u.a. alle Blöcke der Diskette als "belegt" oder "freigegeben" gekennzeichnet. Sie befindet sich auf Spur 18 der Diskette.

1.1.4 Einige Daten der VC 1541-Diskette:

Diskette:

Anzahl Spuren:	35
Anzahl Sektoren je Spur:	17 bis 21 (je nach Größe der Spur)
Bytes (Zeichen) je Block:	256
Gesamtzahl der Blöcke:	683
Zahl der freien Blöcke:	644 (die Directoy belegt den Rest)
Einträge in der Directory:	144 pro Diskette

Laufwerk:

- intelligentes Peripheriegerät mit eigenem Prozessor und eigenem Betriebssystem
- Anschluß am seriellen IEC-Bus von CBM 64 oder VC-20, Gerätenummer 4-15 (standard 8)

Diese Daten sollen vorerst genügen. Sie lernen später weitere kennen und verstehen.

1.2 Das Speichern von Programmen auf Diskette

Die Überlegenheit des Diskettenlaufwerks als externes Speichermedium gegenüber dem Rekorder zeigt sich bereits bei der Abspeicherung von Programmen. Die Speicherung von Programmen ist mit einem Diskettenlaufwerk erheblich komfortabler als mit einem Cassettenrekorder. Ein wesentlicher Vorteil liegt in der Geschwindigkeit der Übertragung von und zum Rechner. Hierzu 2 Beispiele:

Das Abspeichern eines 3 KByte großen Programms dauert:
- mit der Datasette VC-1530 75 Sekunden
- mit der Floppy VC-1541 nur 12 Sekunden

Das Laden eines 16 KByte großen Programms dauert:
- mit der Datasette VC-1530 330 Sekunden
- mit der Floppy VC-1541 nur 50 Sekunden

Ein weiterer Vorteil ist, daß auf der Diskette mehrere Programme übersichtlich abgelegt werden können. Um ein Programm zu laden, schaut man sich lediglich das Inhaltsverzeichnis (Directory) der Diskette an und wählt dann das gewünschte Programm aus. Zwar können auf einer Kassette auch mehrere Programme gespeichert werden, jedoch ist das Aufsuchen eines Programms sehr umständlich, da die entsprechende Bandposition erst durch Spulen der Kassette aufgesucht werden muß. Bevor Sie die Beispiele in den folgenden Abschnitten ausprobieren, sollten Sie beachten, daß vorher eine Diskette gemäß Abschnitt 1.3.2 zu formatieren ist, um Programme auf dieser Diskette abspeichern zu können.

1.2.1 SAVE - Speichern von BASIC-Programmen

Vieleicht waren Sie vorher im Besitz einer Datasette, mit der Sie sicher Programe abgespeichert haben. Der Befehl zur Speicherung von Programmen auf der Floppy-Station unterscheidet sich nicht wesentlich davon. Sie müssen dem Rechner lediglich mitteilen, daß er das Programm auf Diskette und nicht auf Kassette zu speichern hat. Dies geschieht durch zusätzliche Angabe der Gerätenummer 8 hinter dem Befehl SAVE. Alle serienmäßig produzierten Laufwerke sind auf diese Adresse hardwaremäßig vorbereitet. Schreiben Sie nun einmal ein kleines BASIC-Programm und speichern es mit dem Befehl

```
SAVE"TEST",8
```

ab. Geben Sie anschließend den Befehl "NEW" ein, damit der BASIC-Speicher gelöscht wird. Im folgende Abschnitt erfahren Sie dann ,wie das Programm wieder zurückgeholt wird.

1.2.2 LOAD - Laden von BASIC-Programmen

Wie beim vorhergehenden Abschnitt ist dieser Befehl bis auf die zusätzliche Angabe der Gerätenummer mit dem Befehl "LOAD" für die Datasette identisch. Laden Sie nun das im vorherigen Abschnitt gespeicherte Programm mit

```
LOAD "TEST",8
```

wieder in den Speicher. Mit dem Befehl "LIST" können Sie den erfolgten Ladevorgang erkennen. Ein eventuell vorher im Speicher befindliches Programm ist aber nun gelöscht, da bei jedem Ladevorgang das Programm ab der Anfangsadresse des BASIC-Speichers abgelegt wird. Es besteht jedoch die Möglichkeit, das im Speicher befindliche Programm zu erhalten, wenn dessen Endadresse als Anfangsadresse gesetzt wird. Dieses "Zusammenfügen" zweier Programme nennt man "MERGE". Eine entsprechnede Routine ist in einem späteren Abschnitt enthalten.

1.2.3 VERIFY - Überprüfen von gespeicherten Programmen

Wenn Sie ein Programm mit dem Befehl 'SAVE' auf Diskette gespeichert haben, so besteht die Möglichkeit zu überprüfen, ob dieses Programm auch richtig abgelegt wurde. Das wird mit dem VERIFY-Befehl realisiert. Der Befehl hat folgendes Format:

```
VERIFY "filename",8
```

Angenommen Sie haben ein Programm mit 'SAVE "TEST",8' gespeichert. Dann befindet sich dieses Programm immer noch im Speicher. Dieses im Speicher befindliche Programm wird dann durch diesen Befehl mit dem tatsächlich abgespeichertem verglichen. Sind beide Programme identisch, so meldet der Rechner "OK".

Probieren Sie es einmal aus, indem Sie einige BASIC-Zeilen schreiben und dann die folgende Befehlsfolge eingeben:

```
SAVE   "TEST.1",8     Programm wird gespeichert
VERIFY "TEST.1",8     Programm wird überprüft
```

Sicher wird der Rechner sich mit "OK" melden, da bei der Diskettenspeicherung sehr selten Fehler auftreten.

1.2.4 SAVE"@:..." - Überschreiben von Programmen

Versuchen Sie jetzt einmal, Ihr kleines Testprogramm nochmals auf der Diskette zu speichern. Beim zweiten Mal meldet der Computer einen "FILE EXISTS"-Fehler und führt den Befehl nicht aus. Das Betriebssystem der Floppy VC-1541 läßt nicht

zu, daß zwei Programme unter gleichem Namen abgespeichert werden. Dies ist auch logisch, denn wie sollte der Computer sonst beim Ladevorgang erkennen, welches Programm von zwei identischen Sie haben möchten.
Nun kann es aber vorkommen, daß Sie ein bereits auf Diskette abgelegtes Programm laden, ändern und wieder abspeichern möchten. Um dies zu realisieren gibt es drei Mögichkeiten:

1. Sie Speichern das Programm unter einem anderen Namen ab

2. Sie Löschen zuerst das alte Programm auf der Diskette und speichern dann das Neue unter dem alten Namen ab

3. Sie verwenden den Zusatz "@:" vor dem Filenamen im SAVE-Befehl

Dieses Zeichen nennt man "Klammeraffe". Es wird sehr oft in der Datenverarbeitung benutzt. Dieses Zeichen und einen Doppelpunkt setzen Sie vor dem Filenamen. Das könnte dann z.B. so aussehen:

```
SAVE "@:TEST",8
```

Vergessen Sie, dieses Zeichen anzugeben, so bringt das Laufwerk die Fehlermeldung "FILE EXISTS", die Sie dann wie im Abschnitt 1.3.3 beschrieben, auslesen können.
Das neue Programm darf den restlichen Speicherplatz der Diskette nicht überschreiten. Wenn Sie verfolgen, wie das DOS das Überschreiben durchführt, werden Sie den Grund dafür erkennen:

1. Einen freien Block als ersten Datenblock des neuen Programms bestimmen und dessen Adresse im Directory-Eintrag des alten Files speichern.

2. Das neue Programm in einen freien Bereich der Diskette speichern.

3. Die Adresse des neuen Files in die Adresse des alten Files übernehmen.

4. Die vom alten File belegten Blöcke als frei kennzeichen.

Da vor dem Freigeben der vom alten File belegten Blöcke das neue File in die freien Blöcke der Diskette gespeichert wird, darf das neue File nicht die freie Diskettenkapazität überschreiten. Sollte jedoch das neue Programm den freien Diskettenspeicher überschreiten, wird der Speichervorgang abgebrochen.

1.2.5 Laden von Maschinenprogrammen

Maschinenprogramme bestehen aus elementaren Befehlen des Prozessors. Sie benötigen den BASIC-Interpreter nicht und

werden auch nicht als BASIC-Programm geladen. Ein Maschinenprogramm wird über die Sekundäradresse 1 zum Rechner übertragen und "absolut" geladen, d.h. ab der in den ersten beiden Bytes des Diskettenfiles enthaltenen Adresse. Ein Beispiel: Der Befehl

```
LOAD "PROFI-MON 64",8,1
```

lädt den Maschinensprache-Monitor absolut. Da dieser Monitor die dezimale Anfangsadresse 49152 hat, wird er anschließend mit dem Befehl "SYS 49152" gestartet. Sollten Sie ein Maschinenprogramm ohne die Sekundäradresse, d.h. wie ein BASIC-Programm laden, so erscheint bei einem anschließendem RUN die Fehlermeldung "SYNTAX ERROR IN ....". Mit dem Befehl "LIST" erscheint dann das Maschinenprogramm als BASIC-Listing, daß natürlich überhaupt keinen Sinn ergibt. Ein Nachteil ist, daß ein BASIC-Programm von einem Maschinenprogramm anhand der Directory nicht zu unterscheiden ist. Beide werden mit dem Filetyp "PRG" gekennzeichnet. Wenn Sie ein Programm auf der Diskette nicht spezifizieren können, so laden Sie es zuerst mit dem Befehl 'LOAD "programm",8'. Sollte nach einem anschließendem 'RUN' die Meldung 'SYNTAX ERROR IN ....' erscheinen und das anschließend aufgelistete Programm nicht als BASIC-Programm zu identifizieren sein, so handelt es sich um ein Maschinenprogramm. Dieses muß dann mit 'LOAD "programm",8,1' geladen werden. Es kann dann aber nicht mit 'RUN' gestartet werden!. Sie müssen erst die Anfangsadresse dieses Programm ermitteln. Dazu können Sie das in diesem Buch enthaltene Programm zur Auflistung aller Fileparameter benutzen. Diese Anfangsadresse ist dann in den meisten Fällen die Startadresse des Programms, das Sie dann mit 'SYS startadresse' aufrufen. Sie können aber auch die Anfangsadresse mit folgender Befehlsfolge ermitteln:

```
10 OPEN 1,8,2,"programmname,S,R"
20 GET#1,X$:IF X$=""THEN X$=CHR$(0)
30 LB=ASC(X$)
40 GET#1,X$:IF X$=""THEN X$=CHR$(0)
50 HB=ASC(X$)
60 CLOSE 1
70 AD=HB*256+LB
80 PRINT"ANFANGSADRESSE:";AD
```

Das Programm zeigt dann die Adresse nach Eingabe von 'RUN' auf dem Bildschirm an. Hier wird also das Programm als sequentielle Eingabedatei eröffnet. Da die ersten beiden Bytes die Anfangsadresse bilden, wird diese mit den beiden GET-Befehlen ausgelesen und entsprechend aufbereitet. Das erste Byte ist das High-Byte und das zweite das Low-Byte der 2-Byte Adresse. Falls Ihnen die Funktion dieser Befehlsfolge unklar ist: Im nächsten Abschnitt wird die Behandlung von sequentiellen Dateien eingehend erklärt.

1.2.6 Speichern von Machinenprogrammen

Maschinenprogramme werden meistens mit einem Maschinensprache-Monitor oder einem Assembler geschrieben und auch von diesen Programmen heraus abgespeichert. Maschinenprogramme können aber auch mit BASIC geschrieben werden, indem die einzelnen Bytes des Programms mit ihrem dezimalen Wert in DATA-Zeilen abgelegt werden. Ein in BASIC mit Hilfe von DATA-Zeilen geschriebenes Maschinenprogramm hätte folgendes Aussehen:

```
10 AA = anfangsadresse
20 EA = endadresse
30 FOR I=AA TO EA
40 READ X
50 POKE I,PEEK(X)
60 NEXT I
80 DATA .....................................
90 DATA .....................................
```

Es muß in diesem Beispiel noch die dezimale Anfangsadresse in Zeile 10 und die Endadresse in Zeile 20 eingesetzt werden. Die dezimalen Werte der einzelnen Bytes des Maschinenprogramms werden jeweils durch Komma getrennt in den DATA-Zeilen angegeben.

Natürlich können Sie auch als einfachsten Weg Maschinenprogramme, wie sie z.B. auch in diesem Buch und in den anderen DATA BECKER BÜCHERN in reichlicher Form finden, in Form des BASIC-Ladeprogramms abspeichern. Allerdings muß dann jeweils vor der Nutzung der entsprechenden Routine diese erst durch Lesen und Ausführen der DATA-Zeilen erzeugt werden, ein etwas umständlicher und zeitraubender Weg. Wesentlich eleganter und zeitsparender ist die Abspeicherung eines in DATA-Zeilen enthaltenen Maschinenprogramms in Form echter Maschinenbefehle, da ein solches "echtes" Maschinenprogramm nach dem Laden ohne umständliches Umsetzen sofort ausgeführt werden kann.
Um ein derartig gespeichertes Programm als Maschinenprogramm auf Diskette abzulegen wird eine Befehlsfolge benutzt, die etwa so aussieht:

```
10 AA = anfangsadresse
20 EA = endadresse
30 OPEN 1,8,1,"programmname"
40 HB=INT(AA/256):LB=AA-HB*256
50 PRINT#1,CHR$(LB);CHR$(HB);
60 FOR I = AA TO EA
70 PRINT#1,CHR$(PEEK(I));
80 NEXT I
90 CLOSE 1
```

Diese Routine setzt voraus, daß das Maschinenprogramm bereits im Speicher des Rechners mit der vorher beschriebenen Routine abgelegt ist. Soll ein in DATA-Zeilen enthaltenes Maschinenprogramm auf Diskette gespeichert werden, so muß

folgende Routine benutzt werden:

```
10 AA = anfangsadresse
20 EA = endadresse
30 OPEN 1,8,1,"programmname"
40 HB=INT(AA/256):LB=AA-HB*256
50 PRINT#,CHR$(LB);CHR$(HB);
60 FOR I = AA TO EA
70 READ X
80 PRINT#1,CHR$(X);
90 NEXT I
100 CLOSE 1
110 DATA .....
120 DATA .....
```

Auch hier müssen noch die Adressen und die DATA-Zeilen eingesetzt werden. Das derartig gespeicherte "echte" Maschinenprogramm wird dann mit dem Befehl 'LOAD "programmname",8,1" eingesetzt, der dann das Maschinenprogramm von der Diskette lädt. Anschließend wird dieses Programm mit 'SYS (anfangsadresse)' gestartet. Maschinenprogramme können auch von einem BASIC-Ladeprogramm geladen und gestartet werden. So ein Ladeprogramm könnte die folgende Form haben:

```
10 IF A=0 THEN A=1:LOAD"programmname",8,1
20 SYS (anfangsadresse)
```

Der IF-Befehl in Zeile 10 verwirrt zunächst. Er muß mit einbezogen werden, weil nach dem Laden eines Programmes immer wieder in Zeile 10 gestartet wird. Wendet man die Befehlsfolge

```
10 LOAD"programmname",8,1
20 SYS (anfangsadresse)
```

an, so würde immer wieder geladen und der SYS nie erreicht. Wird aber die Variable A auf eins gesetzt, so verzweigt das Programm nach dem erneuten Ablauf von Zeile 10 nach Zeile 20. Dieses Ladeprogramm wird dann zusammen mit dem Maschinenprogramm auf der Diskette abgelegt. Zum Starten des Maschinenprogramms geben Sie nur die Befehle

```
LOAD"ladeprogramm",8
RUN
```

ein. Dies hat den Vorteil, daß die Anfangsadresse des Maschinenprogramms nicht zum Starten benötigt wird, weil das Ladeprogramm den SYS beinhaltet.

1.3 Die Floppy-Systembefehle

Wie schon erwähnt, ist die Floppy VC-1541 wie die großen CBM Floppys CBM 4040, 8050 und 8250 ein intelligentes Peripheriegerät mit einem eigenem Prozessor und einem eigenem Betriebssystem. Dieses eigene Betriebssystem, das DOS (Disk Operating System) belegt k e i n e n Platz im Speicher Ihres VC-20 oder COMMODORE 64 und bietet trotzdem eine Reihe sehr leistungsfähiger Befehle, die den Befehlssatz Ihres COMMODORE Computers wesentlich erweitern. Eine weitere Besonderheit neben der Speicherplatzersparnis (bei fast allen anderen Computern wird das DOS in den Hauptspeicher geladen und belegt dort wertvollen Platz) ist die Tatsache, daß die Befehle des Floppy DOS von der Floppy völlig selbstständig ausgeführt werden, ohne daß Ihr Computer hiermit belastet wird. Da diese Befehle aber nicht im Befehlssatz Ihres VC-20 oder COMMODORE 64 enthalten sind, müssen sie auf eine besondere Art und Weise zur Floppy übertragen werden. Dort rufen diese Befehle dann entsprechende Unterprogramme auf, die die gewünschte Aufgabe durchführen.

1.3.1 Die Befehlsübermittlung zur Floppy-Station

Sämtliche Befehle, die an die Floppy-Station (an das DOS) gerichtet sind, werden über einen "Kanal" gesendet. Dieser Kanal ist der Kanal Nummer 15. Die Datenübertragung über diesen Kanal erfolgt folgendermaßen:

- Öffnen des Kanals (OPEN)
- Datenübertragung (PRINT)
- Schließen des Kanals (CLOSE)

Im OPEN-Befehl muß neben der Kanalnummer noch die Nummer des Gerätes, zu dem die Daten gesendet werden sollen und die logische Fielnummer enthalten sein. Beachten Sie nun die Syntax des OPEN-Befehls zur Übertragung von Floppy-Systembefehlen:

```
OPEN #lfn,8,15,"befehl"
```

Die 8 in dem Befehl adressiert die Adresse der Floppy-Station und der Befehlskanal ist 15. Der Parameter 'lfn' ist die logische Filenummer des OPEN-Befehls, die benötigt wird, um die Übertragungsbefehle (PRINT#,INPUT#,GET$) den OPEN-Befehlen zuzuordnen. Sie ist frei wählbar (1-127). Der Floppy-Systembefehl kann entweder direkt dem OPEN-Befehl folgen, oder aber mit einem PRINT-Befehl nach dem Eröffnen übermittelt werden. Bis zum Schließen dieses Kanals kann eine beliebige Anzahl Systembefehle übertragen werden, die sich natürlich auf die im OPEN-Befehl angegenbene logische Filenummer beziehen müssen.

1.3.2 NEW - Formatieren von Disketten

Der Befehl zum Formatieren lautet "NEW" und kann wie jeder andere Befehl durch sein erstes Zeichen (N) abgekürzt werden. Wie bereits erwähnt, kann der Befehl im OPEN-Befehl oder nachfolgend in einem PRINT-Befehl angegeben werden. Der NEW-Befehl hat folgendes Format:

```
NEW:diskettename,id
```

Der Diskettename umfasst maximal 16 Zeichen und ist im Kopf des Directorys enthalten. Das Identifizierungsmerkmal (ID) der Diskette besteht aus zwei beliebige Zeichen, an der das DOS erkennt, ob eine andere Diskette eingelegt wurde. Da Sie dieses Idenfikationsmerkmal frei wählen können, bietet es sich gut für die Unterscheidung sonst völlig identischer Disketten an, oder aber für eine allgemeine Klassifizierung Ihrer Disketten. Wer nicht mehr als 99 Disketten hat, kann seine Disketten sehr schön an Hand des Identifikationsmerkmals ordnen.
Nun aber ein Beispiel zum Formatieren einer Diskette:

```
OPEN 1,8,15,"NEW:TESTDISKETTE,KL"
```

Geben Sie diesen Befehl nun einmal ein, nachdem Sie eine "rohe" Diskette eingelegt haben. Sie werden feststellen, daß das Laufwerk nun mit dem Formatieren beginnt. Dieser Vorgang dauert ca. 80 Sekunden. Da das Laufwerk mit seinem eigenen Prozessor formatiert und den Prozessor des Rechners nicht benötigt, kann während dem Vorgang weiter mit dem Rechner gearbeitet werden. Der Befehl kann aber auch abgekürzt werden:

```
OPEN 1,8,15,"N:TESTDISKETTE,KL"
```

Soll der Befehl mit einem PRINT übermittelt werden, so muß folgende Befehlsfolge eingegeben werden:

```
OPEN 1,8,15          zum Öffnen des Kanals
PRINT#1,"N:TESTDISKETTE,KL"
```

Die Nummer 1 des PRINT-Befehls bezieht sich auf die logische Filenummer des OPEN-Befehls. Ist der Befehl dem ersten Beispiel entsprechend abgesetzt worden, so können mit dem PRINT-Befehl weitere Befehle über diesen Kanal übermittelt werden.Sollen keine weiteren Befehle übermittelt werden, so muß der Kanal geschlossen werden. Das geschieht mit dem CLOSE-Befehl. Geben Sie nun nach dem Formatieren folgenden Befehl ein:

```
CLOSE 1
```

Nun ist der Befehlskanal geschlossen. Die 1 bezieht sich wieder auf die logische Filenummer des entsprechenden OPEN-Befehls.

1.3.3 Auslesen des Fehlerkanals

Wie Ihnen sicher bekannt ist, gibt der Rechner bei nicht ordnungsgemäßer Programmierung Fehlermeldungen aus. Da die Diskettenbefehle aber nicht von dem Prozessor des Rechners, sondern von dem des Laufwerks überprüft und ausgeführt werden, kann der Rechner die Fehlermeldungen des Laufwerks nicht anzeigen. Fehlermeldungen werden vom Anwender an der aufblinkenden roten Leuchtdiode am Laufwerk erkannt. Um jedoch festzustellen, welcher Fehler aufgetreten ist, muß der Rechner den Kanal 15, über dem die Fehler übermittelt werden, auslesen. Dazu muß der Kanal 15 vom Rechner geöffnet werden, falls dies nicht bereits geschehen ist. Danach wird mit dem INPUT-Befehl die Fehlermeldung ausgelesen. Sie besteht aus 4 Feldern:

1. Feld: Nummer des Fehlers (numerisch)
2. Feld: Bezeichnung des Fehlers (alphabetisch)
3. Feld: Spur (numerisch)
4. Feld: Sektor (numerisch)

Die Spur- und Sektorangabe bezeichnet, wo der Fehler lokalisiert wurde. Diese vier Felder der Fehlermeldung müssen in 4 Variable eingelesen werden, wobei die 2. Variable eine Stringvariable sein muß. Dem INPUT-Befehl müssen dann also 4 Variablen folgen. Ein Beispiel zur Auslesung des Fehlerkanals:

```
OPEN 1,8,15        (falls noch nicht erfolgt)
INPUT#1,FF,FB$,SP,SE
CLOSE 1
```

Da der INPUT-Befehl aber nicht direkt eingegeben werden kann, muß der Fehler innerhalb eines Programms ausgelesen werden. D.h. die oben genannte Befehlfolge muß mit Zeilennummern versehen und dann mit RUN gestartet werden. Das sieht dann z.B. so aus:

```
10 OPEN 1,8,15
20 INPUT#1,FF,FB$,SP,SE
30 PRINT FF;FB$;SP;SE     (zur Anzeige auf dem
                           Bildschirm)
40 CLOSE 1
```

Um die Wirkungsweise dieses Programms zu erkennen, verursachen Sie bitte folgenden Fehler:

```
OPEN 1,8,15,"NEW TESTDISKETTE,T1"
CLOSE1
```

Wenn Sie diese Befehlsfolge eingegeben haben, blinkt die rote Leuchtdiode an dem Floppy-Laufwerk. Haben Sie den Fehler erkannt? Es fehlt der Doppelpunkt nach dem Befehl "NEW". Geben Sie nun die Befehlsfolge zum Auslesen des Fehlerkanals ein und starten mit RUN. Auf dem Bildschirm erscheint dann die Meldung:

```
 34 SYNTAX ERROR 0  0
```

Die 34 ist die Nummer des Fehlers, dessen Klartext dann folgt. Das Feld Spur und Sektor ist 0, weil dieser Fehler diese Angaben nicht benötigt.
Sollte ohne daß ein Fehler aufgetreten ist, der Fehlerkanal ausgelesen werden, so wird die Meldung

```
        0 OK 0 0
```

ausgegeben.
Falls während der Arbeit mit der Floppy-Station die rote Leuchtdiode blinken sollte, so überprüfen Sie erst Ihren Befehl, denn meistens ist der Fehler wie beim o.g. Beispiel leicht zu erkennen. Andernfalls lesen Sie einfach den Fehlerkanal aus. Eine detailierte Beschreibung aller Fehlermeldungen und ihrer Ursachen erfolgt im Abschnitt 1.6.

1.3.4 LOAD "$",8 - Laden des Directory

Das Directory ist das Inhaltsverzeichnis der Diskette. Hier sind alle Files (Programme und Dateien) der Diskette katalogisiert. Beachten Sie unbedingt, daß das Laden der Directory den Verlust eines eventuell vorher im Speicher befindlichen Programms zur Folge hat. Das Directory wird mit

```
        LOAD "$",8
```

geladen und kann dann mit dem LIST-Befehl aufgelistet werden. Probieren Sie es nun einmal mit der dem Laufwerk beigefügten Test/Demo-Diskette aus. Legen Sie diese Diskette in das Laufwerk und geben Sie den o.g. Befehl zum Laden der Directory ein. Danach listen Sie mit dem Befehl LIST das Directorry auf. Es erscheint dann wie folgt auf dem Bildschirm: (Bitte beachten Sie, daß nicht alle VC-1541 mit der derselben Test/Demo-Diskette geliefert werden, da COMMODORE auch hier manchmal nicht angekündigte Änderungen vornimmt).

```
0     "1541test/demo    " zx 2a
13    "how to use"        prg
5     "how part two"      prg
4     "vic-20 wedge"      prg
1     "c-64 wedge"        prg
4     "dos 5.1"           prg
11    "copy/all"          prg
4     "disk addr change"  prg
4     "dir"               prg
6     "view bam"          prg
4     "check disk"        prg
14    "display t&s"       prg
9     "performance test"  prg
5     "sequential file"   prg
13    "random fial"       prg
```

Diesem Directory sind viele Informationen zu entnehmen. Sehen wir uns die 1. Zeile, den Kopf des Directory, einmal an. Das Zeichen '0' in dieser Zeile hat keine besondere Bedeutung. Daneben ist der Name und die ID der Diskette angegeben, wie es bei der Formatierung vereinbart wurde. Die Zeichen '2A' symbolisieren das Diskettenformat. Ist dieses Format nicht '2A', so ist diese Diskette auch nicht auf dieser Art Laufwerk formatiert worden und auch nicht lauffähig.
Nun folgen die einzelnen Files mit Ihrer Blocklänge am Anfang und dem Filetyp am Ende der Zeile. Auf dieser Diskette erkennen Sie 3 verschiedene Filetypen die im Folgenden erklärt werden. Auf die restlichen Filetypen wird später noch eingegangen.

PRG	Dies sind PROGRAM-FILES, d.h. Programme in BASIC oder Maschinensprache
SEQ	So werden sequentielle Dateien gekennzeichnet, die später beschrieben werden
REL	Dies ist eine andere Form der Datenspeicherung, die ebenfalls später beschrieben wird.

Die Länge der Files ist in Blöcken angegeben, von denen jeder 256 Bytes umfasst.So kann man leicht die Größe eines Programms ermitteln. Man muß lediglich von den 256 Bytes eines jeden Blocks 2 Bytes abrechnen, die zur Verkettung der einzelnen Blocks benötigt werden.
Am Ende des Directory ist dann noch die Anzahl der noch freien Blöcke der Diskette ersichtlich. Wenn Sie die Länge der Files aufaddieren und die freien Blöcke hinzuzählen, so resultiert daraus die Gesamtzahl der belegbaren Blöcke auf einer Diskette (664).
Wenn Sie einen Drucker besitzen, so kann dieses Directory wie ein Programmlisting ausgedruckt werden. Dazu verwenden Sie folgende Befehlsfolge:

OPEN 1,4	Öffnen des Druckers
CMD 1	die Bildschirmausgabe wird auf dem Drucker gelenkt
LIST	das Directory wird auf dem Drucker ausgegeben
PRINT#1	ein RETURN wird zum Drucker gesendet
CLOSE 1	der Drucker wird wieder geschlossen

Voraussetzung für den Ausdruck mit dieser Befehlsfolge ist natürlich, daß das Directory mit 'LOAD "$",8' geladen wurde. Sollte sich im Speicher ein BASIC-Programm befinden, so kann ebenfalls mit dieser Routine das Programm ausgedruckt werden. Durch Einsatz des Jokers können Sie bewirken, daß nicht stets das gesamte Directory geladen wird, sondern nur der Teil, der Sie interessiert, z.B. alle Programme. Näheres hierzu in Kapitel 1.3.10

1.3.5 SCRATCH - Löschen von Files

Natürlich muß die Möglichkeit bestehen, nicht mehr benötigte Files zu löschen. Dazu ist der Befehl 'SCRATCH' vorgesehen. Bevor dieser Befehl angewandt wird, sollte man sich stets überzeugen, daß der im Scratch-Befehl angegebene Name auch mit dem des zu löschenden Files übereinstimmt. Ein unabsichtlich gelöschtes File kann die Arbeit von mehrerern Stunden oder sogar Tagen zunichte machen.
Zum Löschen eines Files muß das folgende Format des Befehls beachtet werden:

```
PRINT#lfn,"SCRATCH: filenamel, filename2,...."
```

Es können also auch mehrere Files mit einem Befehl gelöscht werden. Wichtig ist die Tatsache, daß dem Floppy-Befehlskanal innerhalb der Anführungszeichen nicht mehr als 40 Zeichen mit einem PRINT-Befehl übermittelt werden können!
Um z.B. ein File mit dem Namen 'TEST' zu löschen, werden folgende Befehle eingegeben:

```
OPEN 1,8,15,"S:TEST"
CLOSE 1
```

Sollte der Kanal 15 bereits geöffnet sein, so genügt ein PRINT-Befehl:

```
PRINT#1,"S:TEST"
```

Es besteht die Möglichkeit, den Inhalt der gesamten Diskette zu löschen. Dazu wird der im Abschnitt 1.3.10 umschriebene "JOKER" (das Zeichen '*') verwendet:

```
PRINT#1,"S:*"
```

Auch hier ist besondere Vorsicht geboten! Überzeugen Sie sich, ob wirklich alle Files gelöscht werden sollen. Dem Fehlerkanal wird die Meldung

```
01 FILES SCRATCHED nn 00
```

übergeben. 'nn' ist die Anzahl der gelöschten Files. Diese Meldung kann mit der im Abschnitt 1.3.3 angegebenen Routine ausgelesen werden.

1.3.6 RENAME - Umbenennen von Files

Um Files einen anderen Namen zu geben wird der Filename im Fileeintrag der Directory geändert. Der Befehl 'RENAME' ist dafür zuständig. Er hat das folgende Format:

```
RENAME:neuer name = alter name
```

Wenn z.B. das File mit dem Namen "TEST" umbenannt werden soll

1.3.8 INITIALIZE - Initialisieren der Diskette

Das DOS benötigt im Diskettenspeicher immer die aktuelle BAM der im Laufwerk befindlichen Diskette. Die BAM ist der Blockbelegungsplan einer Diskette. Sie kennzeichnet jeden Block als frei oder belegt und wird bei jedem Befehl, der Blöcke belegt oder freigibt aktualisiert. Wenn nun die Diskette gewechselt wird, so erkennt das DOS diesen Vorgang an der unterschiedlichen ID der Diskette. Sollte nun die neue Diskette die gleiche ID haben wie die vorher im Laufwerk befindliche Diskette, so nimmt das DOS den Diskettenwechsel nicht war. Die noch im Speicher befindliche BAM der ersten Diskette ist dann nicht mehr identisch mit der BAM der nächsten Diskette. Befehle, die nicht vor Ausführung die BAM in den Speicher lesen (z.B. alle Direktzugriffs-Befehle) benutzen die im Diskettenspeicher befindliche (falsche!) BAM zur Lokalisierung der belegten bzw. nicht belegten Blöcke.

Deshalb sollte beim Formatieren der Diskette die ID immer unterschiedlich sein. Es ist also nicht sinnvoll jeder Diskette die gleiche ID zu geben. Mit dem Befehl 'INITIALIZE' kann die BAM "von Hand" in den Diskettenspeicher übertragen werden. Dieser Befehl hat folgendes Format:

```
PRINT#lfn,"INITIALIZE"
```

oder abgekürzt

```
PRINT #lfn,"I"
```

Beispiel:

```
OPEN 1,8,15,"I"
CLOSE 1
```

Sollten Sie also für sich selbst oder für andere Programme erstellen, die Datenspeicherung und Diskettenwechsel beinhalten, so empfehlen wir dringend, aus Sicherheitsgründen in Ihrem Programm nach jedem Diskettenwechsel den INITIALIZE-Befehl zu verwenden.

1.3.9 VALIDATE - "Aufräumen der Diskette

Der Befehl 'VALIDATE' gibt alle als belegt gekennzeichneten Blöcke der Diskette, die nicht einem ordnungsgemäß geschlossenen File zuzuordnen sind, wieder frei. Wenn Sie z.B ein File mit 'OPEN' öffnen, Daten übertragen und dieses File aber nicht wieder mit 'CLOSE' schließen, so wird es beim 'VALIDATE' wieder gelöscht. Oder aber Sie arbeiteten mit Direkt-Zugriffs-Befehlen auf der Diskette, beschreiben also Blöcke oder kennzeichnen sie als belegt. Diese Blöcke sind dann keinem File zugeordnet und werden durch diesen Befehl wieder freigegeben.
Der Befehl hat auch noch eine weitere Funktion: Wenn ein File mit "SCRATCH" gelöscht wird, so wird nur der Filetyp im

ersten Byte des Fileeintrags auf "0" gesetzt. Es erscheint somit nicht mehr in der Directory. Wenn Sie nun dieses Byte wieder gemäß dem alten Filetyp erneuern, was entweder mit dem in diesem Buch enthaltenen DOS-Monitor, oder aber auch mit Direkt-Zugriffs-Befehlen durchgeführt werden kann, so regeneriert ein anschließender 'VALIDATE' dieses File. Es ist also wieder im alten Zustand auf der Diskette enthalten.
Der Befehl hat folgendes Format:

```
PRINT#lfn,"VALIDATE"
```

oder in der Kurzform

```
PRINT#lfn,"V"
```

Ein Beispiel:

```
OPEN 1,8,15,"V"
CLOSE 1
```

Sollte Sie einmal eine Diskette besitzen, bei der die aufaddierten Filelängen in Blöcken nicht plus den angegebenen freien Blöcken nicht der Gesamtblockzahl der Diskette (664) entspricht, so stellt der VALIDATE-Befehl wieder den alten Zustannd her.
Ein weiteres Beispiel: Wenn Sie ein Programm oder eine Datei speichern wollen, daß den freien Diskettenspeicher überschreitet, so meldet das DOS den Fehler "DISK FULL". Wenn die Diskette auch vorher noch einige freie Blöcke aufwies, so ist die Anzahl der freien Blöcke nun Null. Mit dem VALIDATE-Befehl werden nun diese, ursprünglich freien Blöcke wieder freigegeben.

1.3.10 ? * - Der "Joker"

Es gibt zwei Jokerzeichen: Den Stern (*) und das Fragezeichen (?). Der Stern an einer bestimmten Stelle des Filenamens symbolisiert, daß das erste File auf der Diskette relevant ist, das mit den Zeichen vor dem Stern beginnt. Ein Beispiel:

```
LOAD "TEST*",8
```

Dieser Befehl lädt das erste Programm, dessen ersten vier Buchstaben "TEST" beinhalten. Der Befehl

```
LOAD "*",8
```

lädt das erste Programm der Diskette, da kein Zeichen vor dem Stern angegeben ist. Der Stern in einem SCRATCH-Befehl hat eine andere Funktion. Hier wird nicht das erste File gelöscht, sondern ALLE. Z.B. löscht der Befehl

```
OPEN1,8,15,"S:TEST*"
CLOSE 1
```

alle Files, die mit den Buchstaben "TEST" beginnen. Dies ist unbedingt zu beachten! Auch das Laden der Directory kann mit dem Stern selektiert erfolgen. Ein beispiel:

```
LOAD "$A*",8
```

lädt nur das Directory mit den Files, die mit dem Buchstaben "A" beginnen.
Das DOS bietet eine weitere Einsatzmöglichkeit des Sterns, die in keiner bisherigen Anleitung zu ersehen ist: Es können auch Filetypen selektiert werden, wenn nach dem Stern ein Gleichheitszeichen mit anschließendem ersten Buchstaben des gewünschten Filetyps angegeben wird. Hier eine Übersicht:

```
*=S       selektiert nur sequentielle Files
*=P       selektiert Programmfiles
*=R       selektiert relative Files
*=U       selektiert User-Files
```

Geben Sie z.B.

```
LOAD "$*=P"
```

ein, so werden nur die Programme auf der Diskette in das Directory übernommen und anschließend mit 'LIST' ausgegeben. Auch können mit dem SCRATCH-Befehl z.B. alle sequentiellen Files auf der Diskette mit folgendem Befehl gelöscht werden:

```
OPEN 1,8,15,"S:*=S"
CLOSE 1
```

Natürlich kann vor diesem Stern auch noch eine Zeichenfolge angegeben werden, sodaß dann nur die sequnetiellen Files gelöscht werden, deren Namen mit dieser Zeichenfolge beginnen.

Mit dem Fragezeichen können im Filenamen Buchstaben an beliebigen Stellen als "nicht relevant" gekennzeichnet werden. Um die Funktion des Fragezeichens zu erläutern, folgen nun zwei Beispiele von abgekürzten Filenamen und ihren Auswirkungen:

```
A????       - fünfstellige Filenamen, deren erster Buch-
              stabe "A" ist, sind angesprochen
????TEST    - achtstellige Filenamen, deren letzen vier
              Buchstaben "TEST" beinhalten, sind ange-
              sprochen
```

Eine Kombination von Stern und Fragezeichen ist erlaubt. Jedoch sollte beachtet werden, daß nach dem Stern weder Buchstaben, noch Fragezeichen folgen, da diese Kombinationen keinen Sinn ergeben. Zwei Beispiele zur Kombination von Stern und Fragezeichen:

```
????.*      - alle Filenamen, die vor dem Punkt vier Buch-
              staben besitzen, sind angesprochen
```

`TEST.??*`	- alle mindestens 7-stellige Filenamen, deren ersten fünf Zeichen "TEST." beinhalten, sind angesprochen.
`TEST-??01*=S`	- alle mindestens 9-stelligen, sequentiellen Files, deren Namen in den ersten 5 Stellen "TEST-" und in den Stellen 6 bis 7 "01" enthalten, sind angesprochen

1.4 Sequentielle Datenspeicherung

Ein Diskettenlaufwerk sollte nicht ausschließlich zur Programmspeicherung genutzt werden. Spätestens dann, wenn Sie eigene Programme schreiben, die eine große Datenmenge zu verwalten haben, werden Sie eine schnelle Datenorganisation benötigen. Die sequentielle Datenspeicherung ist zwar nicht die schnellste, aber die einfachste Methode, Daten zu verwalten, was gerade für Anfänger wichtig sein dürfte. Diese Datenorganisation ist vergleichbar mit der sequentiellen Datenspeicherung auf Kassette, die ebenfalls in dieser logischen Reihenfolge in ein Programm integriert wird:

1. Laden des Programms

2. Laden der kompletten Daten in den Speicher des Rechners

3. Verwalten der Daten im Speicher (ändern, löschen, hinzufügen)

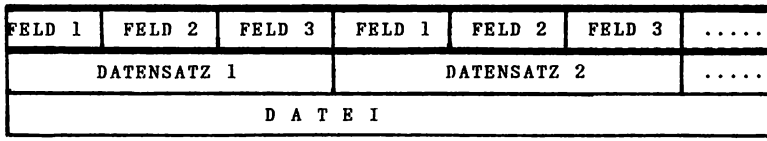
4. Speichern der aktuellen Daten auf einem externen Speichermedium (Kassette, Diskette)

5. Verlassen des Programms

Es ist selbstverständlich, daß die maximale Datenmenge von der Größe des Speichers im Rechner abhängig ist, da ein Datensatz in einer sequentiellen Datei nicht direkt auf der Diskette oder Kassette geändert oder gelöscht werden kann. Dazu muß die gesamte Datei eingelesen, geändert und wieder abgespeichert werden. Das Laden und Speichern der Datei geschieht bei Einsatz eines Diskettenlaufwerkes wesentlich schneller als bei einem Kassettenlaufwerk. Dies ist der erste Vorteil der Datenspeicherung mit Diskette.
Der zweite Vorteil ist, daß zum Anfügen eines Datesatzes an eine sequentiellen Diskettendatei nicht die gesamte Datei eingelesen werden muß. Hierzu wird die Datei zum Anfügen (APPEND) geöffnet. Dies ist bei der Speicherung auf Kassette nicht möglich.
Erwähnenswert ist noch, daß Programme, die bisher Daten sequentiell auf Kassette verwalteten, auf einfache Art und Weise an Diskettenspeicherung angepasst werden können. Hierzu müssen nur die entsprechenden OPEN-Befehle geändert werden.

1.4.1 Das Prinzip

Eine sequentielle Datei besteht aus mehreren Datensätzen, die wiederum in Felder aufgeteilt sind. Am Beispiel einer Adressen-Datei ist dies leicht zu verdeutlichen: Die einzelnen Adressen stellen die Datensätze dieser Datei dar. Ein Adressen-Satz besteht aus mehreren Felder (Name, Vorname, usw). Die Struktur einer Datei läßt sich etwa so darstellen:



Die Datensätze einer Datei sind wie die Felder innerhalb dieses Datensatzes hintereinander (sequentiell) angeordnet. Die Felder und somit auch die Datensätze dürfen unterschiedlich lang sein. So kann z.B. das Feld 1 des Datensatzes 1 länger sein als das Feld 1 des Datensatzes 2. Dies ist möglich, da die Felder voneinander durch ein Zeichen (RETURN) getrennt werden, die von dem PRINT-Befehl erzeugt und von dem INPUT-Befehl erkannt werden. Jedem Feld ist eine Variable zugeordnet, die mit einem PRINT-Befehl geschrieben und mit einem INPUT-Befehl eingelesen wird. Es besteht aber auch die Möglichkeit mit einem Befehl einen ganzen Datensatz zu lesen oder schreiben. Da setzt aber voraus, das alle Datensätze die gleiche Länge haben, da innerhalb des Programms diese Datensätze mit Hilfe von speziellen Befehlen in Felder zerlegt werden müssen. Dazu muß dem Programm die genaue Position jedes Feldes im Datensatz bekannt sein.
Doch wie erkennt der Rechner beim Einlesen der Daten, wann ein Feld bzw. Datensatz beendet ist? Dazu wird hinter jedem Feld der Datei ein 'RETURN' gesetzt, das die einzelnen Felder voneinander trennt. Dieses 'RETURN' hat im ASCII-Code den dezimalen Wert 13. Am Beispiel einer Telefon-Datei wird dies sichtbar: Unsere Telefon-Datei soll aus 3 Feldern bestehen:

FELD 1 : NAME
FELD 2 : VORNAME
FELD 3 : TELEFONNUMMER

Schauen wir uns einen Ausschnitt aus dieser bereits beschriebenen Datei nun an (das Zeichen '+' symbolisiert ein 'RETURN'):

```
Zeichen:          1111111111222222222233333333334444444445
          12345678901234567890123456789012345678901234567890
          ------------------------------------------------------
Datei:    KUNZE+HANS+236+KURZ+TIM+1213+SCHULZE+UTE+65432+...
          ------------------------------------------------------
```

Es ist zu erkennen, daß die Felder ungleich lang sind und jeweils durch ein 'RETURN' getrennt sind. Dieses 'RETURN' wird bei der Übertragung mit einem PRINT-Befehl jeweils hinter den Daten gesetzt, sofern diesem PRINT-Befehl kein Semikolon, das ein 'RETURN' unterdrückt, folgt. Mit einem INPUT-Befehl werden diese Daten dann in eine Variable übernommen und zwar bis zum 'RETURN'. Danach muß ein weiterer INPUT-Befehl folgen, um das nächste Feld zu lesen, usw. Die folgenden Abschnitte erläutern alles, was zur Erstellung von Programmen mit sequentieller Datenspeicherung erforderlich ist.

1.4.2 Das Eröffnen einer sequentiellen Datei

Um eine Datei zu erstellen, muß sie vorher geöffnet werden. Beim Öffnen zum Beschreiben wird Folgendes durchgeführt:

1. Es wird geprüft, ob auf der Diskette bereits ein File mit diesem Namen existiert. Wenn ja, wird die Fehlermeldung "FILE EXISTS" ausgegegeben.

2. Der entsprechende Fileeintrag in der Directory wird angelegt. Dabei wird im Filetyp gekennzeichnet, daß dieses File noch nicht geschlossen ist, was dann in der aufgelisteten Directory durch einen Stern vor dem Filetyp ersichtlich ist.

3. Es wird ein freier Block gesucht, auf dem die ersten Daten gespeichert werden. Die Adresse (Spur und Sektor) wird im Fileeintrag gespeichert.

4. Die Anzahl der Blocks im File wird auf 0 gesetzt, da noch kein Block dieses Files beschrieben ist.

Nach dem Erstellen der Datei kann diese dann geändert oder erweitert werden. Im OPEN-Befehl wird festgelegt, zu welchem Zweck die Datei geöffnet werden soll. Das Format des OPEN-Befehls sieht folgendermaßen aus:

```
OPEN lfn,8,sa,"filename,filetyp,modus"
```

Die logische Filenummer liegt zwischen 1 und 127, wenn nach einem PRINT-Befehl auf dieses File nur 'RETURN' gesendet werden soll. Dies wird in der Regel der Fall sein. Ist die logische Filenummer größer als 127 (128-255), so sendet der PRINT-Befehl nach jedem 'RETURN' noch einen 'LINE-FEED' (Zeilenvorschub). Dies ist zum Beispiel bei Druckern notwendig, die nach einem 'RETURN' keinen automatischen Zeilenvorschub geben.
Die Sekundäradresse kann einen Wert zwischen 2 und 14 annehmen und für Ein- und Ausgaben verwendet werden. Sie bezeichnet den Kanal des Floppy-Laufwerkes, über den die Daten übertragen werden sollen. Die Sekundäradresse 0 und 1 ist vom Betriebssystem zum Speichern und Laden von Programmmen reserviert. Sekundäradresse 15 ist für den Befehls-und Fehlerkanal bestimmt. Sollten mehrere Dateien gleichzeitig geöffnet sein, so muß neben der logischen Filenummer unbedingt die Sekundäradresse unterschiedlich sein, da immer nur ein Kanal für eine Datei zuständig sein kann. Wird jedoch eine Datei mit der Sekundäradresse geöffnet, mit der vorher bereits eine Datei geöffnet wurde, so wird die erste Datei geschlossen.
Was oft mißachtet wird, ist die Tatsache, daß maximal 3 Kanäle mit jeweils einer Datei geöffnet werden können. Zu Verwaltung von relativen Dateien benötigt das DOS jedoch 2 Kanäle gleichzeitig. Demnach sind folgende Maximalkombinationen möglich:

- 1 relative und 1 sequentielle Datei
- 3 sequentielle Dateien

Bei der Angabe des Filenamens ist darauf zu achten, daß dieser Filename nicht bereits auf der Diskette existiert. Soll eine Datei zum Schreiben geöffnet werden, die bereits auf der Diskette existiert, so muß wie bei dem Befehl 'SAVE' dem Filenamen der Klammeraffe mit dem anschließendemm Doppelpunkte vorangestellt werden!. Ein Beispiel:

```
OPEN 1,8,2,"@:ADRESSEN,S,W"
```

Bei der Eröffnung der Datei muß der Filetyp angegeben werden. Diese Filetypen werden im OPEN-Befehl wie folgt angekürzt:

S - sequentielles File
U - User-File
P - Programmfile
R - relatives File

User-Files sind sequentielle Files, die jedoch in der Directory als USR-File ausgewiesen werden. Es sind keine Dateien im eigentlichen Sinne. Dieser Filetyp wird gerne benutzt, wenn Ausgaben, die normalerweise auf dem Bildschirm erfolgen (BASIC-Listing, Directory) zur Floppy "umgeleitet" werden. Im Kapitel 1.4.6 finden Sie eine Beschreibung dieser Methode.

Der letzte Parameter (modus) legt fest, wie der Datenkanal genutzt werden soll. Es gibt vier Möglichkeiten:

W - Schreiben einer Datei (WRITE - Kapitel 1.4.3)
R - Lesen einer Datei (READ - Kapitel 1.4.4)
A - verlängern einer sequentiellen Datei

(APPEND - Kapitel 1.4.4)
M - Lesen einer nicht geschlossenen Datei
(wurde von uns im DOS-Listing "entdeckt" und wird im Kapitel 1.4.5 erläutert)

Öffnen Sie nun einmal eine sequentielle Datei mit dem Namen "SEQU.TEST" zum Schreiben:

```
OPEN 1,8,2,"SEQU.TEST,S,W"
```

Wenn Sie anschließend mit 'LOAD "$",8' das Directory laden und mit 'LIST' ausgeben, werden Sie feststellen, daß dieses File mit einem Stern vor dem Filetypen als geöffnet gekennzeichnet ist:

```
0   SEQU.TEST         *SEQ
```

Diese Datei läßt sich nun aber nicht mehr schließen! Bevor also nach dem Eröffnen und Beschreiben einer Datei das Directory geladen wird, muß unbedingt das File geschlossen werden!

Während eine Datei geöffnet ist, darf zwar der Befehls/Fehlerkanal 15 geöffnet werden, jedoch hat das Schließen des Kanals 15 zur Folge, daß alle anderen Files auch geschlossen werden. Dies sollten Sie unbedingt beachten.
Nun einige Beispiele zum OPEN-Befehl:

```
OPEN 1,8,2,"SEQU.TEST,S,R"      - ein sequentielles File wird
                                  zum Lesen geöffnet
OPEN 2,8,3,"SEQU.TEST,U,W"      - ein User-File wird zum
                                  Schreiben geöffnet
OPEN 3,8,4,"TEST,P,R"           - ein Program-File wird zum
                                  Lesen geöffnet
OPEN 4,8,5,"SEQU.TEST,S,A"      - ein sequentielles File wird
                                  zum Anfügen von Daten ge-
                                  öffnet
OPEN 5,8,6,"KUNDEN.1983,S,M"    - Die Kundendatei wurde nicht
                                  ordnungsgemäß geschlossen
                                  und soll gelesen werden.
```

1.4.3 Datenübertragung Floppy/Rechner

Nach dem Eröffnen eines Files zum Schreiben können die darin zu speichernden Daten an die Floppy mit dem PRINT-Befehl übermittelt werden. Dieser Befehl überträgt zusätzlich ein 'RETURN', das zum Trennen der Daten benötigt wird. Im folgenden Beispiel wird eine Datei eröffnet, beschrieben und wieder geschlossen. Da der PRINT-Befehl auch direkt, d.h. außerhalb eines Programms eingegeben werden kann, lassen sich die entsprechenden Befehle hintereinander absetzen und ausführen. Eröffnen Sie nun ein File mit dem Namen "TEST.1":

```
OPEN 1,8,2,"TEST.1,S,W"
```

Sie werden bemerkt haben, daß die rote Leuchtdiode an dem Floppy-Laufwerk aufleuchtet. Sie signalisiert, daß ein File geöffnet ist. Die Datei kann nun beschrieben werden. Hier wird z.B. ein Datensatz einer Adressendatei, bestehend aus 4 Feldern erstellt:

```
PRINT#1,"HANS"
PRINT#1,"SCHULTZ"
PRINT#1,"KASTANIENSTR. 7"
PRINT#1,"4000 DÜSSELDORF"
```

Nun sind diese Daten in die Datei aufgenommen worden, und das File kann wieder mit 'CLOSE 1' geschlossen werden. Die rote Leuchtdiode ist gleichzeitig erloschen. Um diese Daten nun wieder zu lesen, muß die Datei im Lese-Modus (R) eröffnet werden. Da der INPUT-Befehl zum Einlesen der Daten nicht direkt eingegeben werden kann, muß ein kleines Programm geschrieben werden:

```
10  OPEN 1,8,2,"TEST.1,S,R"
```

```
20  INPUT#1,VN$
30  INPUT#1,NN$
40  INPUT#1,ST$
50  INPUT#1,OR$
60  CLOSE 1
70  PRINT"VORNAME:   ";VN$
80  PRINT"NAME:      ";NN$
90  PRINT"STRASSE:   ";ST$
100 PRINT"PLZ/ORT:   ";OR$
```

Das Programm ist einfach zu erklären:

Zeile 10 Die Datei "TEST.1" wird zum Lesen geöffnet

Zeile 20-50 Die Daten werden in der selben Reihenfolge eingelesen, in der sie vorher geschrieben wurden. Es werden dazu Variablen genutzt, die nachher zum Ausgeben der Daten benötigt werden.

Zeile 60 Die Datei wird wieder geschlossen.

Zeile 70-100 Die Daten werden mit entsprechendem Begleittext auf dem Bildschirm ausgegeben.

Wenn Sie diese Befehlsfolge eingegeben und mit 'RUN' gestartet haben, erscheinen die Daten, die vorher in die Datei geschrieben wurden, nun auf dem Bildschirm:

```
VORNAME:    HANS
NAME:       SCHULTZ
STRASSE:    KASTANIENSTR. 7
PLZ/ORT:    4000 DÜSSELDORF
```

Zum Einlesen der Daten wurden 4 INPUT-Befehle eingesetzt, da eine Adresse aus 4 Feldern besteht. Wenn aber z.B. eine Datei gespeichert werden soll, deren Datensätze aus ca. 20 Feldern bestehen, so ist es sehr aufwendig, zum Einlesen 20 INPUT-Befehle ins Program aufzunehmen. Durch Programmieren einer Schleife kann dies wesentlich vereinfacht werden. Am Beispiel unseres kleinen Programms ist dies ersichtlich:

```
10 OPEN 1,8,2,"TEST.1,S,R"
20 FOR I=1 TO 4
30 INPUT#1,D$(I)
40 NEXT I
50 CLOSE 1
60 PRINT"VORNAME:   ";D$(1)
70 PRINT"NAME:      ";D$(2)
80 PRINT"STRASSE:   ";D$(3)
90 PRINT"PLZ/ORT:   ";D$(4)
```

Hier wurden nicht 4 Stringvariablen, sondern eine indizierte Variable mit dem Index 1-4 benutzt. Es ist zu beachten, daß der Index beim BASIC 2.0 höchstens 10 betragen darf, wenn er nicht mit einer DIM-Anweisung höher definiert wurde. Soll in

unserem Beispiel ein Datensatz mit 20 Felder eingelesen werden, so muß vorher die Anweisung 'DIM D$(20)' gegeben werden.
Es gibt noch eine weitere Möglichkeit der verkürzten Ein- und Ausgabe von Daten: Mit dem INPUT-Befehl zur Dateneingabe von Tastatur können mehrere Variablen, die durch ein Komma getrennt sind, eingegeben werden. Z.B:

```
INPUT VN$,NN$,TE
```

Bei diesem Befehl müssen drei Variablen z.B. folgendermaßen eingegeben werden:

```
NORBERT,MÜLLER,7465
```

Die eingelesenen Daten werden dann mit

```
PRINT VN$,NN$,TE
```

wieder auf dem Bildschirm ausgegeben werden. Auf diese Weise können auch Daten in eine sequentielle Datei geschrieben und auch wieder eingelesen werden. Der einzige Unterschied ist, daß beim Schreiben in eine Datei Stringvariablen durch ein in Hochkomma eingeschlossenes Komma getrennt werden müssen. Wenn z.B. die o.g. Variablen in eine Datei geschrieben werden sollen, muß der PRINT-Befehl folgendermaßen geändert werden:

```
PRINT#1,VN$","NN$","TE
```

Numerische Varialen werden nur mit dem Komma von anderen Variablen getrennt. Zum Einlesen der Daten wird dann der Befehl

```
INPUT#1,VN$,NN$,TE
```

eingesetzt. Da die maximal einzugebene Zeichenzahl mit einem INPUT-Befehl 88 nicht überschreiten darf, ist diese Schreibweise nur begrenzt einsatzfähig. Sollte ein Feld in einem Datensatz länger als 88 Zeichen sein, so muß ein anderer Befehl zum Einlesen benutzt werden. Dies ist der GET-Befehl, der jedes Zeichen einzeln einliest. Angenommen Sie möchten einen Datensatz lesen, der aus einem Feld mit der Länge von 100 Zeichen besteht. Dieser Satz kann dann mit folgender Routine in eine Stringvariable übernommen werden:

```
10 OPEN 1,8,.................
20 D$=""
30 FOR I=1 TO 100
40 GET#1,X$
50 D$=D$+X$
60 NEXT I
70 GET#1,X$
80 CLOSE 1
```

Nach Ablauf dieser Befehlsfolge enthält die Stringvariable den 100 Zeichen umfassenden Datensatz. Nach dem Öffnen einer

sequentiellen Datei wird vom DOS ein Zeiger eingerichtet, der immer auf das Zeichen zeigt, das hinter den bisher gelesenen Daten liegt. Da wir annehmen, daß der 100-Zeichen umfassende Datensatz mit einem PRINT-Befehl ohne abschließendes Semikolon in die Datei geschrieben wurde, wurde der Datensatz mit einem RETURN abgeschlossen. Nach dem Lesen des 100. Zeichens weist der Zeiger auf dieses RETURN. Der nächste GET-Befehl in Zeile 70 ist also notwendig, um das 'RETURN', das sich hinter dem Datensatz befindet, zu lesen. Dadurch erhält der erste GET-Befehl zum Lesen des nächsten Satzes wieder das erste Zeichen und nicht das 'RETURN'.

In diesem Beispiel sind wir von einer konstanten Datensatzlänge von 100 Zeichen ausgegangen. In der Regel ist die Datensatzlänge einer sequentiellen Datei aber nicht konstant. Derartge Dateien müssen also, falls die maximale Datensatzlänge die INPUT-Grenze von 88 Zeichen überschreitet, mit einer GET-Schleife gelesen werden, die das trennende RETURN als Satzende erkennt. Eine derartige Routine sieht dann so aus:

```
10 OPEN 1,8,.....................
20 S$=""
30 GET#1,X$
40 IF X$=CHR$(13)THEN 80
50 S$=S$+X$
60 IF ST<>64 THEN 30
70 CLOSE1:END
80 PRINT S$
90 GOTO 20
```

Hier wird eine Datei mit variabler Satzlänge gelesen und auf dem Bildschirm ausgegeben. Anstatt der Ausgabe auf Bildschirm können diese Datensätze natürlich auch anders verarbeitet werden.
Wenn Sie diese Probleme, die bei einer Datensatzlänge von über 88 Zeichen auftreten, vermeiden wollen, so teilen Sie den Datensatz in mehrere Teile auf, die Sie dann nach dem Einlesen wieder zusammenfügen.

1.4.4 Anhängen von Datensätzen

Stellen Sie sich vor, Sie müßten zum Erweitern einer sequentiellen Datei diese komplett in den Hauptspeicher laden, erweitern und wieder in der erweiterten Form abspeichern. Es wäre sicher sehr zeitraubend. Aus diesem Grunde bietet das DOS eine komfortable Möglichkeit, einer sequentieller Datei Daten anzuhängen, ohne die Datei vorher einzulesen. Dies ermöglicht der Eröffnungsmodus 'A' (APPEND). Wenn Sie also eine sequentielle Datei erstellt haben, wie z.B. im vorherigem Abschnitt, so können Sie immer wieder Daten anhängen, indem Sie im OPEN-Befehl den Modus 'A' angeben. Ein Beispiel:

Geben Sie folgende Befehlsfolge ein:

```
OPEN 1,8,2,"TEST.2,S,W"
PRINT#1,"1. DATENSATZ"
CLOSE1
```

Sie haben nun eine sequentielle Datei mit einem Datensatz erstellt. Diese Datei soll nun mit folgender Befehlsfolge um 2 Datensätze erweitert werden:

```
OPEN 1,8,2,"TEST.2,S,A"
PRINT#1,"2. DATENSATZ"
PRINT#1,"3. DATENSATZ"
CLOSE1
```

Nun enthält die Datei 'TEST.2' 3 Datensätze. Mit dem folgenden Programm können Sie dies überprüfen:

```
100 OPEN 1,8,2,"TEST.2,S,R"
110 FOR I=1 TO 3
120 INPUT#1,DS$
130 PRINTDS$
140 CLOSE 1
```

Nach dem Starten dieses Programms werden die Datensätze ausgelsen und auf dem Bildschirm angezeigt.
Sie haben erkannt, daß der Modus 'A' bei sequentieller Datenorganisation eine schnelle Erweiterung der Datei ermöglicht.

1.4.5 Schließen einer sequentiellen Datei

Mit dem CLOSE-Befehl werden geöffnete Dateien wieder geschlossen. Dieser Befehl hat das Format

```
CLOSE lfn
```

Der Parameter 'lfn' bezieht sich auf die logische Filenummer der Datei, die bei dem entprechendem OPEN-Befehl angegeben wurde. Sollen mehrere Dateien abgeschlosssn werden, so muß für jede Datei ein CLOSE-Befehl abgesetzt werden. Mit dem Schließen der letzten Datei erlischt die rote Leuchtdiode am Laufwerk wieder.
Wie Ihnen bereits bekannt ist, werden die Daten über einen Kanal zur Floppy gesendet. Dieser Kanal ist ein floppyinterner Speicher (Puffer genannt), in dem die vom Rechner übermittelten Daten zunächst zwischengespeichert werden. Erst wenn dieser Puffer gefüllt ist, werden die darin befindlichen Daten auf die Diskette geschrieben.
Beim Schließen der Datei werden die noch im Puffer befindlichen Daten auf die Diskette geschrieben. Eine nicht geschlossene Datei ist also nicht vollständig, und wird auch vom Disketten-Betriebssystem als nicht ordnungsgemäß geschlossenees File gekennzeichnet. Das DOS erlaubt nun im

Modus 'R' (READ) auf diese Datei keinen Lesezugriff mehr und meldet "WRITE FILE OPEN".
Nun wäre es aber sehr ärgerlich, wenn das DOS keinen Lesezugriff auf diese Datei zulassen würde. Aus diesem Grunde bietet das DOS den Modus 'M'. Eine in diesem Modus geöffnete Datei, die als nicht ordnungsgemäß geschlossene Datei gekennzeichnet ist, kann so gelesen werden. Sinnvoll ist es, die gelesenen Datesätze in eine zweite Datei zu schreiben und diese dann natürlich ordnungsgemäß zu schließen. Auf diese Weise kann man Dateien "retten".
Das folgende Programm bietet die Möglichkeit, eine nicht geschlossene Datei (Ursprungsdatei) in eine korrekt geschlossene Datei (Zieldatei) zu übertragen:

```
100 INPUT"URSPRUNGSDATEI";U$
110 INPUT"ZIELDATEI";Z$
120 OPEN 1,8,2,U$+",S,M"
130 OPEN 2,8,3,Z$+",S,W"
140 INPUT#1,X$
150 PRINT#2,X$
160 IF ST<>64 THEN 140
170 CLOSE 1:CLOSE 2
180 OPEN 1,8,15,"S:"+U$
190 CLOSE 1
```

Am Ende des Programms wird dann die nicht mehr benötigte Ursprungsdatei gelöscht.

1.4.6 "Umleiten" der Bildschirmausgabe

Jede Ausgabe, die auf dem Bildschirm erfolgt (PRINT,LIST usw.) kann als sequentielle Datei auf die Diskette umleitet werden. Dies wird mit dem CMD-Befehl erreicht, der folgendes Format hat:

```
CMD lfn
```

Dazu muß zuerst ein File eröffnet werden, das zur Unterscheidung von sequentiellen Dateien den Filetyp "USR" erhält. Soll z.B. das Listing eines BASIC-Programms als sequentielles File auf Diskette gespeichert werden, so dient dazu die folgende Befehlsfolge:

```
OPEN 1,8,2,"TEST.LIST,U,W"
CMD 1
LIST
PRINT#1
CLOSE 1
```

Der Befehl 'CLOSE 1' bewirkt gleichzeitig, daß die weitere Ausgabe wieder auf dem Bildschirm erfolgt.
Das Speichern von Programmen als sequentielle Dateien auf Diskette ist z.B. sehr nützlich, wenn man ein Programmlisting mit einer Textverarbeitung lesen möchte, um es in Text mit

einzubauen. Voraussetzung ist, daß die entsprechende Textverarbeitung in der Lage ist, in ASCII-Code gespeicherte Dateien zu lesen.
So sind übrigens die Listings in diesem Buch vom Commodore 64 der Textverarbeitung SUPERSCRIPT auf einem Commodore 8032 übergeben worden.
Um dieses File nun wieder auf dem Bildschirm auszugeben, benötigen Sie die folgende Routine:

```
10 OPEN 1,8,2,"TEST.LIST,U,R"
20 GET#1,X$
30 PRINT X$;
40 IF ST<>64 THEN 20
50 CLOSE 1
```

Diese Routine ist eine Schleife, die jedes Zeichen (Byte) des Files liest und auf dem Bildschirm ausgibt. Das Ende des File wird an der Statusvariablen ST erkannt, die bei Fileende auf 64 gesetzt wird. Zur Ausgabe des sequentiellen Files ist folgende Befehlsfolge erforderlich:

```
10 OPEN 1,8,2,"TEST.LIST,U,R"
20 OPEN 2,4
30 GET#1,X$
40 PRINT#2,X$;
50 IF ST<>64 THEN 30
60 CLOSE 1:CLOSE 2
```

Hier wurde zusätzlich der Drucker geöffnet, der die Geräteadresse 4 besitzt.

1.4.7 Sequentielle Datei als Tabelle im Rechner

Sequentielle Dateien müssen zur Datenverwaltung komplett im Rechner vorhanden sein. Dazu wird meist eine zweidimensionale Tabelle benutzt. Diese Tabelle nennt man auch Matrix, da durch Angabe von zwei Koordinaten jedes beliebige Feld eines Datensatzes adressiert werden kann. Dazu verwendet man eine zweifach indizierte Variable, die mit einer DIM-Anweisung reserviert werden muß. Der erste Index bezeichnet den Datensatz, der zweite Index das Feld innerhalb dieses Datensatzes. Das folgende Schaubild zeigt das Beispiel einer Tabelle:

	Feld 1	Feld 2	Feld 3
Datensatz 1	D$(1,1)	D$(1,2)	D$(1,3)
Datensatz 2	D$(2,1)	D$(2,2)	D$(2,3)
Datensatz 3	D$(3,1)	D$(3,2)	D$(3,3)
Datensatz 4	D$(4,1)	D$(4,2)	D$(4,3)
Datensatz 5	D$(5,1)	D$(5,2)	D$(5,3)
Datensatz 6	D$(6,1)	D$(6,2)	D$(6,3)

Diese Tabelle zeigt eine Datei, die aus 6 Datensätzen mit je 3 Feldern besteht. Als Variable wurde D$ benutzt, die mit 'DIM D$(6,3)' reserviert wird. Um eine sequentielle Datei als Tabelle in den Rechner einzulesen, ist es erforderlich, eine solche Datei mit z.B. 6 Datensätzen a' 3 Feldern zu erzeugen. Dazu benutzen Sie das folgende Programm:

```
100 OPEN 1,8,2,"TESTFILE.SEQ,S,W"
110 FOR X=1 TO 6
120 PRINT CHR$(147);
130 PRINT"DATENSATZ ";X
140 PRINT"-------------"
150 FOR Y=1 TO 3
160 PRINT"FELD ";Y;": ";
170 INPUT X$
180 PRINT#1,X$
190 NEXT Y
200 NEXT X
210 CLOSE 1
```

Hier wird eine zweifach verschachtelte Schleife verwendet, mit deren Variablen die Datensätze und -felder numeriert werden. Geben Sie nun die 6 Datensätze ein. Nach Beendigung dieses Programms befinden sich diese 6 Datensätze als sequentielle Datei auf der Diskette. Ein Tip: Speichern Sie dieses Programm mit 'SAVE"TEST.INP",8 ab, damit Sie es jedezeit wieder laden können.
Diese Datei soll nun als Tabelle in den Rechner eingelesen werden. Dazu dient ebenfalls eine zweifach verschachtelte Schleife, deren Variablen nun zur Indizierung der Tabellenplätze benötigt werden:

```
100 OPEN 1,8,2,"TESTFILE.SEQ,S,R"
110 DIM D$(6,3)
120 FOR X=1 TO 6
130 FOR Y=1 TO 3
140 INPUT#1,D$(X,Y)
150 NEXT Y
160 NEXT X
180 CLOSE 1
```

Nach dieser Befehlsfolge befinden sich die Daten in der mit D$ bezeichneten Tabelle. Mit einem PRINT-Befehl können Sie nun überprüfen, ob die Daten an richtiger Stelle gespeichert wurden. Da jedes Feld mit den Indizes adressierbar ist, geben Sie z.B. 'PRINT D$(1,2)' ein, um das 2. Feld des 1. Datensatzes auf dem Bildschirm anzuzeigen. Sinnvoll ist es nun, die Felder eines ausgewählten Datensatzes anzeigen zu lassen. Benutzen Sie dazu die folgende Routine, nachdem Sie das vorherige Programm abgespeichert haben:

```
100 INPUT"NUMMER DES DATENSATZES: ";X
120 PRINT"--------------------------"
130 PRINT"FELD 1: ";D$(X,1)
140 PRINT"FELD 2: ";D$(X,2)
150 PRINT"FELD 3: ";D$(X,3)
```

Sie haben sicher erkannt, daß der erste Index (die Satznummer) nach der Abfrage als Variable in jeder Feldausgabe verwendet wird. Der zweite Index (die Feldnummer) ist dann jeweils konstant.
Diese Tabelle kann nun beliebig geändert werden. Fügen Sie dem o.g. Programm die folgenden Zeilen an:

```
160 PRINT"--------------------------"
170 INPUT"ZU ÄNDERNDES FELD:      ";Y
180 INPUT"NEUER INHALT:           ";D$(X,Y)
190 PRINT"O.K."
200 PRINT"WEITERE ÄNDERUNGEN (J/N)?"
210 GET X$:IF X$=""THEN 210
220 IF X$="J"THEN 100
230 IF X$="N"THEN END
240 GOTO 210
```

Hier wird die Nummer des zu ändernden Feldes als zweiter Index benutzt, der dann neben dem bereits ausgewählten Index des Datensatzes zur Eingabe des neuen Tabellenplatzes eingesetzt wird.
Diese geänderte Tabelle muß nun wieder auf die Diskette gespeichert werden. Benutzen Sie dazu die folgende Routine. Speichern Sie aber vorher die Änderungsroutine ab!

```
100 OPEN 1,8,2,"@:TESTFILE.SEQ,S,W"
110 FOR X=1 TO 6
120 FOR Y=1 TO 3
130 PRINT#1,D$(X,Y)
140 NEXT Y
150 NEXT X
160 CLOSE 1
```

Auch diese Routine ist durch Anwendung einer zweifach verschachtelten Schleife relativ kurz. Der sogenannte Klammeraffe vor dem Filenamen ist notwendig, da das bereits existierende, alte File überschrieben werden soll.
Der Datenzugriff ist bei dieser Tabelle sehr schnell. Die Zugriffszeit ist von der Tabellengröße unabhängig. Jedoch ist die Größe der Tabelle und somit die Datenmenge abhängig von

der Speicherkapazität. Der große Speicher des Commodore 64 wird mit der Tabellenverarbeitung bestens ausgenutzt. Angenommen, Sie haben ein Programm zur Verwaltung von Adressen geschrieben, das vieleicht um die 8 KByte umfasst, so verbleiben noch 30 Kbyte zur Speicherung der Adressen. Wenn man bedenkt, daß zur Speicherung einer Adresse ca. 80 Zeichen notwendig sind, so können Sie immerhin 384 Adressen ständig im Speicher verwalten! Und das mit einer Zugriffszeit, die selbst bei der raffiniertesten Dateiorganisation (indexsequentiell, relativ) nicht zu übertreffen ist. Bei großen Datenmengen ist die sequentielle Speicherung jedoch nicht mehr anwendbar.

1.4.8 Suchen in der Tabelle

Wie bei der Tabellenverarbeitung erwähnt, kann jeder Datensatz einer Tabelle indiziert werden. Da die Tabelle zweidimensional ist, stellt der erste Index die Nummer des Datensatzes dar, mit der jeder beliebige Satz adressiert und ausgegeben werden kann. Wenn in einer als Tabelle im Hauptspeicher geladenen Datei ein Satz manipuliert werden soll, so setzt das voraus, daß der Anwender die Nummer dieses Satzes kennt. Diese Nummer kann im einfachsten Fall z.B. die Artikel- oder Kundennummer sein. Es gibt aber auch Dateien, die kein geeignetes Feld zur Durchnumerierung der Daten enthalten. In derartigen Dateien muß der gewünschte Datensatz in der Tabelle gesucht werden. Dazu müssen alle Datensätze der Tabelle durchsucht, und mit dem als Suchbegriff eingegebenen Feld verglichen werden. Ein praktisches Beispiel dazu:

Erstellen Sie zuerst mit folgendem Programm eine Datei, die zum Beispiel Namen und Telefonnummern speichert:

```
100 OPEN 1,8,2,"TELEDAT,S,W"
110 PRINT CHR$(147)
120 INPUT "NAME      :";NN$
130 INPUT "VORNAME   :";VN$
140 INPUT "VORWAHL   :";VW$
150 INPUT "NUMMER    :";NU$
160 PRINT "EINGABE KORREKT (J/N)?"
170 GET X$:IF X$=""OR X$ <>"J" AND X$<>"N" THEN 170
180 IF X$="N"THEN110
190 PRINT#1,NN$","VN$","VW$","NU$
200 PRINT"WEITERE EINGABEN (J/N)?"
210 GET X$:IF X$=""OR X$<>"J" AND X$ <>"N" THEN 210
220 IF X$="N"THEN 240
230 GOTO 110
240 CLOSE 1
```

Die Dokumentation des Programms:

Zeile 100 Die sequentielle Datei "TELEDAT" wird zum Schreiben geöffnet

Zeile 110	Der Bildschirm wird gelöscht
Zeile 120-150	Die 4 Felder der Datei werden von Tastatur eingegeben
Zeile 160-180	Falls die Daten nicht korrekt eingegeben wurden, kann die Eingabe wiederholt werden
Zeile 190	Die 4 Felder des Datensatzes werden hintereinander in die Datei geschrieben
Zeile 200-220	Hier kann die Eingabe und das Programm beendet werden
Zeile 230	Die Eingabe wird fortgesetzt
Zeile 240	Die in Zeile 100 geöffnete Datei wird geschlossen

Geben Sie nun dieses Programm ein, starten es und erfassen einige Daten. Speichern Sie dieses Testprogramm auf Diskette, wenn Sie es später einmal zusammen mit den folgenden Beispielen zu einem Programm zusammenfassen möchten. Im letzten Abschnitt dieses Kapitels finden Sie jedoch das komplette Programm zur Verwaltung Ihres Telefonregisters.
Wenn Sie nun einige Daten erfasst haben, möchten Sie vieleicht die ein oder andere Telefonnummer ausfindig machen. Dazu können Sie u.U. die gesamte Datei auf Bildschirm oder Drucker ausgeben und die entsprechende Telefonnummer heraussuchen. Dies ist jedoch eine aufwendige Methode, besonders dann, wenn die Datei viele Datensätze umfasst.
Die Suche nach der Telefonnummer eines bestimmten Namens kann man dem Rechner überlassen. Er durchläuft in einer Schleife die Datensätze und vergleicht sie mit dem gewünschten Namen. Danach gibt er dann den gesamten Datensatz, in dem dieser Name enthalten ist, aus. Die folgende Routine arbeitet dementsprechend:

```
100 OPEN 1,8,2,"TELEDAT,S,R"
110 DIM D$(100,4):X=1
120 INPUT#1,D$(X,1),D$(X,2),D$(X,3),D$(X,4)
130 IF ST<>64 THEN X=X+1:GOTO 120
140 CLOSE 1
150 PRINT CHR$(147)
160 PRINT"GESUCHTER NAME: ";S$
170 FOR I=1 TO X
180 IF D$(I,1)=S$ THEN 210
190 NEXT I
200 PRINT "NAME NICHT GEFUNDEN!":GOTO 280
210 PRINT "NAME GEFUNDEN:"
220 PRINT "---------------"
230 PRINT "NAME:         ";D$(I,1)
240 PRINT "VORNAME:      ";D$(I,2)
250 PRINT "VORWAHL:      ";D$(I,3)
260 PRINT "NUMMER:       ";D$(I,4)
270 PRINT "---------------"
```

```
280 PRINT "WEITER (J/N)?"
290 GET X$:IFX$=""OR X$<>"J"AND X$<>"N"THEN 290
300 IF X$="J"THEN 150
310 PRINT"PROGRAMM BEENDET":END
```

Die Dokumentation zu dem Programm:

Zeile 100	Die sequentielle Datei "TELEDAT" wird zum Lesen geöffnet
Zeile 110	Die Tabelle wird für 100 Datensätze dimensioniert und der Index auf 1 gesetzt
Zeile 120	Die Datensätze werden in die Tabelle eingelesen
Zeile 130	Die Statusvariable ST, die bei Dateiende 64 enthält wird geprüft. Liegt kein Dateiende vor, so wird der Index um 1 erhöht und erneut eingelesen
Zeile 140	Die in Zeile 100 geöffnete Datei wird geschlossen
Zeile 150	Der Bildschirm wird gelöscht
Zeile 160	Der zu suchende Name wird von Tastatur eingelesen und in die Variable S$ gespeichert
Zeile 170-190	Die Schleife sucht in der Tabelle den Datensatz, dessen Namensfeld mit dem gesuchten Namen übereinstimmt. Ist der Satz gefunden, so wird zur Ausgaberoutine verzweigt.
Zeile 200	Der Name wurde nicht gefunden.
Zeile 210-270	Der Satz, der den gesuchten Namen enthält, wird komplett ausgegeben
Zeile 280-310	Es wird die Möglichkeit eingeräumt, erneut einen Namen zu suchen

Sie werden feststellen, daß dieser Suchvorgang selbst bei einer größeren datenmenge recht schnell ist, da die Datei vor dem Suchen als Tabelle in den Rechner geladen wurde. Das Suchen innerhalb des Speichers im Rechner ist schneller, als die Suche auf der Diskette, wenn die Datei sich im Speicher befindet. Das Programm läßt sich leicht derartig abändern, daß nicht nur nach dem Namen, sondern nach einem beliebigen anderen Feld gesucht wird. Vieleicht versuchen Sie es einmal, eine Telefonnummer zu suchen.

Das eben behandelte Programm bricht die Suche nach dem ersten Datensatz, der dem Suchbegriff entspricht, ab. Das ist aber

nicht immer sinnvoll. Wenn z.B. in der erstellten Telefondatei alle Datensätze gesucht und ausgegeben werden sollen, die einer bestimmten Vorwahl entsprechen, so ist eine andere Routine notwendig. Diese Routine muß nach dem Auffinden eines Datensatzes diesen ausgeben und die Suche fortsetzen. Das folgende Programm erfüllt diese Anforderungen:

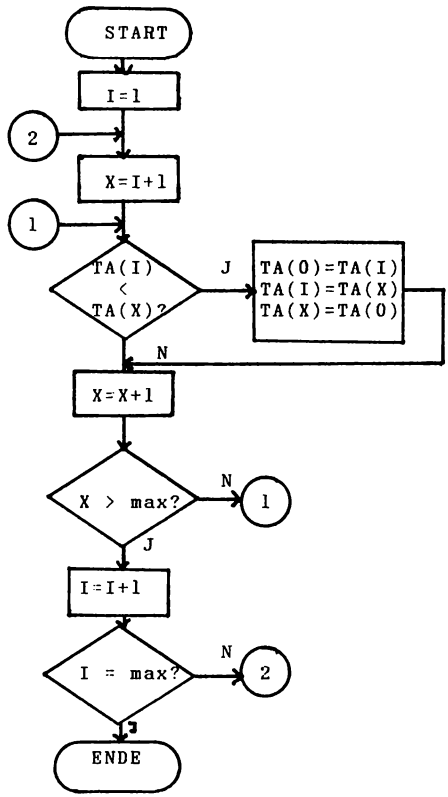
```
100 OPEN 1,8,2,"TELEDAT,S,R"
110 DIM D$(100,4):X=1
120 INPUT#1,D$(X,1),D$(X,2),D$(X,3),D$(X,4)
130 IF ST<>64 THEN X=X+1:GOTO 120
140 CLOSE 1
150 PRINT CHR$(147)
160 PRINT"GESUCHTE VORWAHL: ";S$
170 FOR I=1 TO X
180 IF D$(I,3)=S$THEN 210
190 NEXT I
200 PRINT"DATEIENDE!":GOTO 300
210 PRINT"---------------"
220 PRINT"NAME:          ";D$(I,1)
230 PRINT"VORNAME:       ";D$(I,2)
240 PRINT"VORWAHL:       ";D$(I,3)
250 PRINT"NUMMER:        ";D$(I,4)
260 PRINT"---------------"
270 PRINT"WEITER (J/N)?"
280 GETX$:IFX$=""OR X$<>"J"AND X$<>"N"THEN 280
290 IF X$="J"THEN 190
300 PRINT"SUCHE BEENDET!":END
```

Hier wird die Suche fortgesetzt, wenn ein Datensatz mit der entsprechenden Vorwahl gefunden wurde. Dies bewirkt die Zeile 290, die das Programm nicht beendet, sondern die Schleife fortsetzt. Erst nach Durchsuchen aller Datensätze meldet das Programm "Dateiende". Wenn Sie den Ablauf dieses Programms vestanden haben, so entwickeln Sie vieleicht einmal eine Suche nach dem Vornamen! Sicher wird es Ihnen mit Zuhilfenahme der o.g. Befehlsfolge keine Schwierigkeiten bereiten.

1.4.9 Einfaches Sortieren der Tabelle

In der Datenverarbeitung ist es oft erforderlich Daten sowohl in numerischer als auch in alphanumerischer Form zu sortieren. Dies war schon immer ein rechenzeitintensiver Vorgang, den Programmierer durch immer wieder verbesserten Sortiermethoden zu verkürzen wußten. Doch gerade in Verbindung mit der Programmiersprache BASIC, die in der Form eines Interpreters doch relativ langsam ist, ist das Sortieren sehr zeitaufwendig.
Warum werden Daten eigentlich sortiert? Stellen Sie sich ein Telefonbuch vor, in dem die Namen völlig ungeordnet enthalten sind. Sie müßten dann von Anfang bis Ende das Telefonbuch durchsuchen, um einen bestimmten Namen zu finden. Die

Sortierung bietet also Vorteile beim Aufsuchen von Daten innerhalb einer Datenmenge. Auch der Computer kann wesentlich schneller in einer sortierten Datei suchen.
Es gibt mehrere Sortiermethoden, die sich hauptsächlich in der Geschwindigkeit unterscheiden. Die einfachste Sortiermethode ist die Methode des Vergleichens eines Tabellenplatzes mit jedem anderen. Soll eine Tabelle aufsteigend sortiert werden, so wird der erste Tabellenplatz mit dem zweiten verglichen. Ist der erste größer, so wird er mit dem zweiten vertauscht. Danach wird der erste Tabellenplatz mit dem dritten verglichen, usw., bis der letzte Platz erreicht ist. Danach befindet sich der kleinste Tabellenplatz am Anfang, also auf dem richtigen Platz. Der nächste Durchlauf berücksichtigt den ersten Tabellenplatz also nicht mehr. An einem Programmablaufplan läßt sich diese Logik verdeutlichen:



Dieses Sortierprogramm geht vom Index 1 aus, der als Anfangsindex in die Variable I gespeichert wird. Der zweite Index ist die Variable X, die den um eins erhöhten Anfangsindex I enthält. Dann wird der erste Tabellenplatz mit dem zweiten verglichen. Ist der Inhalt von TA(I) größer als der von TA(X), muß das Programm deren Inhalt über das Hilfsfeld TA(0) vertauschen. Dieser Ringtausch verhindert, daß die Inhalte der beiden Felder verlorengehen. Danach wird der Index X um eins erhöht, also auf den Wert 3 gebracht, wonach dann der erste Tabellenplatz TA(I) mit dem dritten TA(X) verglichen wird, usw. Wenn der letzte Tabellenplatz erreicht ist (X > letzter Index), befindet sich im ersten Tabellenplatz TA(I) der kleinste Tabellenplatz und der Index I wird um eins erhöht. Nun wird der zweite Tabellenplatz mit allen weiteren verglichen, usw.
Diese Sortiermethode erscheint auf den ersten Blick recht umständlich. Die Vergleiche laufen im Hauptspeicher aber relativ schnell ab. Für kleinere Sortiermengen reicht diese Methode aus.
Um dieses Programm laufen zu lassen, muß erst eine Tabelle aufgebaut werden. Wir benutzen eine Tabelle mit 12 Plätzen, die alphanumerische Daten (Strings) enthält. Diese Tabelle wird mit folgender Routine gefüllt:

```
100 DIM TA$(12)
110 FOR I=1 TO 12
120 INPUT TA$(I)
130 NEXT I
```

Nach Starten dieser Befehlsfolge geben Sie 12 beliebige Strings ein, die dann mit dem folgendem Programm aufsteigend sortiert werden:

```
140 I=1
150 X=I+1
160 IF TA$(I) < TA$(X) THEN 180
170 TA$(0)=TA$(I):TA$(I)=TA$(X):TA$(X)=TA$(0)
180 X=X+1
190 IF X <= THEN 160
200 I=I+1
210 IF I <> 12 THEN 150
220 FOR I=1 TO 12
230 PRINT TA$(I)
240 NEXT I
```

Die Tabelle wird nun sortiert und auf dem Bildschirm ausgegeben. Soll anstatt dieser eindimensionalen Tabelle eine zweidimensionale Tabelle wie unsere im Speicher befindliche Telefondatei sortiert werden, so müssen alle Felder eines Satzes vertauscht werden. Die Zeilen 160 - 170 werden zum Sortieren nach Namen folgendermaßen abgeändert:

```
160 IF D$(I,1) < D$(X,1) THEN 180
170 D$(0,1)=D$(I,1):D$(I,1)=D$(X,1):D$(X,1)=D$(0,1)
171 D$(0,2)=D$(I,2):D$(I,2)=D$(X,2):D$(X,2)=D$(0,2)
172 D$(0,3)=D$(I,3):D$(I,3)=D$(X,3):D$(X,3)=D$(0,3)
```

```
173 D$(0,4)=D$(I,4):D$(I,4)=D$(X,4):D$(X,4)=D$(0,4)
```

Einen größeren Datenbestand dieserart zu sortieren ist sehr zeitaufwendig. Wenn Sie auch bei größeren Datenmengen auf eine schnelle Sotierung angewiesen sind, so empfehlen wir Ihnen die sehr schnelle Maschinensprache-Sortierroutine aus unserem Buch "64 TIPS UND TRICKS"

1.4.10 ADRESSENVERWALTUNG mit sequentieller Datenspeicherung

Zum Ende dieses Kapitels bieten wir Ihnen eine komfortable Adressenverwaltung, die wahrscheinlich jeder Anwender sinnvoll einzusetzen weiß. Dieses Pogramms ist gleichzeitig eine Anregung zur Erstellung vieler Dateiverwaltungen.
Eine Adressensatz dieses Programms besteht aus folgenden Feldern:

- ANREDE
- NAME 1
- NAME 2
- STRASSE/NR.
- PLZ/ORT
- TELEFON
- BEMERKUNG

Die Anwendung der Felder 'NAME 1' und 'NAME 2' bleibt dem Anwender überlassen. So kann z.B. in 'NAME 1' der Vorname und in 'NAME 2' der Zuname gespeichert werden. Oder aber in 'NAME 1' die Firma und in 'NAME 2' "zu Händen...". Das Feld 'BEMERKUNG' kann z.B. die Adressen gruppieren (Familie, Beruf, Freunde usw.).

Das Programm bietet nach dem Starten folgende Auswahlmöglichkeiten:

```
-1- DATEI LADEN
-2- DATEI SICHERN
-3- DATEN EINGEBEN
-4- DATEN AENDERN
-5- DATEN SELEKTIEREN/AUSGEBEN
-6- DATEN LOESCHEN
-0- PROGRAMM BEENDEN
```

-1- DATEI LADEN

Nach Auswahl dieses Unterprogramms muß der Dateiname der zu ladenen Datei eingegeben werden. Falls die Datei auf der Diskette existiert wird diese geladen. Dann wird auf dem Bildschirm die Anzahl der Datensätze, die die Datei enthält, ausgegeben. Sollte während dem Laden ein Fehler auftreten, oder die Datei gar nicht existieren, so wird die Meldung "DISKETTENFEHLER!" ausgegeben. Nach Abschluß des Unterprogramms mit 'RETURN' erscheint wieder das Auswahlmenü.

-2- DATEI SICHERN

Falls Sie eine Datei nach dem Laden geändert oder erweitert haben, so müssen Sie vor Beendigung des Programms die Datei mit diesem Unterprogramm auf Diskette sichern. Als Dateiname wird hier entweder der Name, der bei der erstmaligen Datenerfassung festgelegt wurde oder der Name der geladenen Datei verwendet. Eine evtl. unter gleichem Namen existierendes File wird überschrieben.
Während der Arbeit mit diesem Programm sollten die Daten zwischendurch immer wieder gesichert werden, da ein Stromausfall die Daten im Rechner löscht. Nach dem Sichern der Daten kann mit dieser Datei wieder weiter gearbeitet werden. Sie muß also nicht erst wieder geladen werden.

-3- DATEN EINGEBEN

Dieses Unterprogramm hat zwei Funktionen:
1. Es wurde noch keine Datei geladen. Bevor die Daten erfasst werden können muß vorher ein Dateiname festgelegt werden. Die nachfolgenden Daten werden dann unter diesem Namen gesichert. Es sollte ein Name angegeben werden, der bisher noch nicht auf der Diskette existiert, da sonst die alte Datei überschrieben wird.

2. Es befinden sich schon Daten im Rechner. Die im Rechner befindliche Datei wird nun erweitert.

Nach der Erfassung einer Adresse erscheint die Meldung "RICHTIG (J/N)?". Hier wird die Möglicheit gegeben, die eingegebenen Daten zu korrigieren. Dazu drücken Sie die Taste 'N'. Sind alle Daten korrekt eingegeben worden, so drücken Sie 'J'. Nun erscheint die Meldung "WEITERE EINGBABEN (J/N)?". Soll die Erfassung fortgesetzt werden, so drücken Sie die Taste 'J'. Wird die Taste 'N' gedrückt, so erscheint wieder das Auswahlmenü.

-4- DATEN AENDERN

Nach Auswahl dieses Unterprogramms muß die zu ändernde Adresse bestimmt werden. Hierzu muß sowohl der Name 1, als auch der Name 2 eingegeben werden. Sind diese beiden Angaben nicht bekannt, so können in dem Unterprogramm "DATEN SUCHEN/SELEKTIEREN" beide Namen aufgesucht werden. Nach Eingabe dieser Namen wird die Adresse in der Datei gesucht. Wird sie gefunden, so erscheint die komplette Adresse mit den numerierten Feldern ausgegeben. Nun muß die dem zu ändernen Feld entsprechende Nummer eingegeben werden. Nun wird der neue Inhalt bestimmt. Die Adresse wird noch einmal im neuen Zustand angezeigt. Sind keine weiteren Änderungen in diesem Satz erforderlich, so wird die Taste '9' gedrückt. Anschließend fragt das Programm, ob eine weitere Adresse

geändert werden soll. Diese Frage wird dann mit den Tasten 'J' und 'N' beantwortet.

-5- DATEN SELEKTIEREN/AUSGEBEN

Dies ist ein sehr komplexes und vielseitiges Unterprogramm. Zuerst bestimmen Sie, ob die selektierten Adressen auf dem Bildschirm (Taste 'B') oder Drucker (Taste 'D') ausgegeben werden sollen. Haben Sie sich für die Ausgabe auf dem Drucker entschieden, so müssen Sie nochmals auswählen, ob die Adressen mit allen Feldern auf normales Druckerpapier (Taste 'P') oder die Felder 1-5 auf Aufklebern (Taste 'A') gedruckt werden sollen. Die Adreßaufkleber müssen einreihig sein und das Format 89 * 36 mm haben.
Zum Selektieren der Daten füllen Sie eine Suchmaske. Bei Feldern, die nicht relevant sind, geben Sie nur 'RETURN'. Wollen Sie z.B. alle Adressen ausgeben, die dem Postleitzahlengebiet 4 entsprechen, so geben Sie in den ersten 4 Felder nur 'RETURN'. Im Feld 'PLZ/Ort' geben Sie die Zahl 4 mit anschließendem 'RETURN' ein. Die restlichen 2 Felder werden ebenfalls mit 'RETURN' übergangen.

Einige Beispiele selektierter Daten:

```
ANREDE       : FIRMA
NAME 1       : 'RETURN'
NAME 2       : 'RETURN'
STRASSE/NR.  : 'RETURN'
PLZ/ORT      : 4000
TELEFON      : 'RETURN'
BEMERKUNG    : 'RETURN'
```

Hier werden alle Firmen ausgegeben, die ihren Sitz in Düsseldorf haben.

```
ANREDE       : 'RETURN'
NAME 1       : M
NAME 2       : 'RETURN'
STRASSE/NR.  : 'RETURN'
PLZ/ORT      : 'RETURN'
TELEFON      : 'RETURN'
BEMERKUNG    : FAMILIE
```

Alle Familienmitglieder, deren Name 1 mit 'M' anfängt, werden ausgegeben.

Sie sehen, wie vielseitig dieses Selektieren ist. Probieren Sie es selbst einmal aus.

-6- DATEN LOESCHEN

Nach Eingabe des 1. und 2. Namens der Adresse wird diese noch

einmal angezeigt. Das Programm fragt, ob diese Adresse wirklich gelöscht werden soll. Esrt nach Betätigung der Taste 'J' wird dann gelöscht.

-0- PROGRAMM BEENDEN

Bevor das Programm beendet wird, wird darauf hingewiesen, daß das Programm mit 'GOTO 110' ohne Datenverlust wieder gestartet werden kann. Das ist wichtig, falls Sie einmal vergessen, die Daten vor Beendigung des Programms zu sichern.

Doch nun das Programm-Listing:

```
100 POKE 53280,5:POKE53281,2:PRINTCHR$(158);:DIMD$(100,7)
110 GOSUB2030
120 PRINT"WAEHLEN SIE DIE GEWUENSCHTE FUNKTION:"
130 PRINT"--------------------------------------":PRINT
140 PRINT"         -1- DATEI LADEN"
150 PRINT"         -2- DATEI SICHERN"
160 PRINT"         -3- DATEN EINGEBEN"
170 PRINT"         -4- DATEN AENDERN"
180 PRINT"         -5- DATEN SELEKTIEREN/AUSGEBEN"
190 PRINT"         -6- DATEN LOESCHEN":PRINT
200 PRINT"         -0- PROGRAMM BEENDEN"
210 PRINT
220 PRINT"             AUSWAHL (0-6)?"
230 GETX$:IFX$<"0"ORX$>"6"THEN230
240 IFX$<>"0"THEN340
250 PRINT:PRINT"             SICHER (J/N)?"
260 GETX$:IFX$<>"N"ANDX$<>"J"THEN260
270 IFX$="N"THEN110
280 GOSUB2030
290 PRINTTAB(9);"DAS PROGRAMM KANN MIT":PRINT
300 PRINTTAB(15);"'GOTO 110'":PRINT
310 PRINTTAB(8);"WIEDER GESTARTET WERDEN,":PRINT
320 PRINTTAB(4);"OHNE DASS DATEN VERLOREN GEHEN!"
330 END
340 ONVAL(X$)GOSUB360,540,680,880,1190,1770
350 GOTO110
360 REM **********
370 REM DATEI LADEN
380 REM **********
390 GOSUB2030
400 INPUT"NAME DER DATEI :";DN$
410 OPEN15,8,15
420 OPEN1,8,2,DN$+",S,R"
430 INPUT#15,FE:IF FE=0THEN460
440 PRINT"DISKETTENFEHLER!!!"
450 GOTO510
460 X=1
470 INPUT#1,D$(X,1),D$(X,2),D$(X,3),D$(X,4),D$(X,5),D$(X,6),
    D$(X,7)
```

```
480 IF ST<>64 THEN X=X+1:GOTO 470
490 PRINT"DATEI IST GELADEN UND BEINHALTET ";X
500 PRINT"DATENSAETZE":PRINT
510 CLOSE1:CLOSE15
520 PRINT"WEITER MIT RETURN"
530 INPUTX$:RETURN
540 REM *************
550 REM DATEI SICHERN
560 REM *************
570 IF X>0THEN590
580 GOSUB2230:RETURN
590 GOSUB2030
600 OPEN1,8,2,"@:"+DN$+",S,W"
610 FORI=1TOX
620 PRINT#1,D$(I,1)","D$(I,2)","D$(I,3)",";
630 PRINT#1,D$(I,4)","D$(I,5)","D$(I,6)","D$(I,7)
640 NEXT
650 PRINT"DATEI IST GESICHERT":CLOSE1:PRINT
660 PRINT"WEITER MIT RETURN"
670 INPUTX$:RETURN
680 REM ***************
690 REM DATEN EINGEBEN
700 REM ***************
710 IFX>0THENGOTO730
720 GOSUB2030:INPUT"DATEINAME ";DN$
730 X=X+1
740 GOSUB2030
750 PRINT"DATENEINGABE:"
760 PRINT"-------------":PRINT
770 I=X:GOSUB2110
780 FORI=1TO7:PRINTCHR$(145);:NEXT
790 FORI=1TO7:PRINTTAB(12);:INPUTD$(X,I):NEXT
800 PRINT:PRINT"RICHTIG (J/N)?"
810 GETX$:IFX$<>"N"ANDX$<>"J"THEN810
820 IFX$="J"THEN840
830 GOTO740
840 PRINT"WEITERER EINGABEN (J/N)?"
850 GETX$:IFX$<>"J"ANDX$<>"N"THEN850
860 IFX$="J"THEN730
870 RETURN
880 REM ***************
890 REM DATEN AENDERN
900 REM ***************
910 IFX>0THEN930
920 GOSUB2230:RETURN
930 GOSUB2030
940 INPUT"NAME 1: ";N1$
950 INPUT"NAME 2: ";N2$
960 FORI=1TOX
970 IFD$(I,2)=N1$ANDD$(I,3)=N2$THEN1010
980 NEXTI
990 PRINT"NAME NICHT GEFUNDEN!"
1000 PRINT"WEITER MIT RETURN":INPUTX$:RETURN
1010 GOSUB2030
1020 PRINT"-1- ANREDE      :";D$(I,1)
1030 PRINT"-2- NAME 1      :";D$(I,2)
```

```
1040 PRINT"-3- NAME 2      :";D$(I,3)
1050 PRINT"-4- STRASSE/NR.:";D$(I,4)
1060 PRINT"-5- PLZ/ORT     :";D$(I,5)
1070 PRINT"-6- TELEFON     :";D$(I,6)
1080 PRINT"-7- BEMERKUNG   :";D$(I,7):PRINT:PRINT
1090 PRINT"NR. DES ZU AENDERNDEN FELDES: ":PRINT"(9=KEINE
     AENDERUNG)";:PRINT
1100 GETX$:IFVAL(X$)<1 OR VAL(X$)>7ANDVAL(X$)<>9THEN1100
1110 IFVAL(X$)=9THEN1150
1120 Y=VAL(X$)
1130 INPUT"NEUER INHALT";D$(I,Y):PRINT
1140 GOTO1010
1150 PRINT"WEITER AENDERUNGEN (J/N)?"
1160 GETX$:IFX$<>"J"ANDX$<>"N"THEN1160
1170 IFX$="J"THEN880
1180 RETURN
1190 REM **************************
1200 REM DATEN SELEKTIEREN/AUSGEBEN
1210 REM **************************
1220 IFX>0THEN1240
1230 GOSUB2230:RETURN
1240 GOSUB2030:PRINT"AUSGABE AUF DRUCKER (D) ODER BILDSCHIRM
     (B)?"
1250 GETX$:IFX$<>"D"ANDX$<>"B"THEN1250
1260 O$=X$:IFO$="B"THEN1300
1270 PRINT:PRINT"PAPIER (P) ODER AUFKLEBERN (A)?"
1280 GETX$:IFX$<>"P"ANDX$<>"A"THEN1280
1290 D$=X$
1300 GOSUB2030
1310 PRINT"GEBEN SIE DIE SUCHBEGRIFFE EIN:"
1320 PRINT"BEI NICHT RELEVANTEN FELDERN NUR RETURN!";
1330 PRINT"----------------------------------------":PRINT
1340 I=0:GOSUB2110
1350 FORI=1TO7:PRINTCHR$(145);:S$(I)="":NEXT
1360 FORI=1TO7:PRINTTAB(12);:INPUTS$(I):NEXT
1370 IFO$="B" OR D$="A"THEN1450
1380 GOSUB2030:PRINT"DRUCKER EINGESCHALTET (J)?"
1390 GETX$:IFX$<>"J"THEN1390
1400 OPEN1,4
1410 PRINT#1,"ANREDE";SPC(4);"NAME 1";SPC(14);"NAME 2";
     SPC(14);"STRASSE"
1420 PRINT#1,SPC(3);"PLZ/ORT";SPC(18);"TELEFON";SPC(8);
     "BEMERKUNG"
1430 FORI=1TO79:PRINT#1,"=";:NEXT:PRINT#1
1440 CLOSE1
1450 FORI=1TOX
1460 FORY=1TO7
1470 IFS$(Y)=LEFT$(D$(I,Y),LEN(S$(Y)))THENZ=Z+1:GOTO1480
1480 NEXTY
1490 IFZ=7THEN GOSUB 1550
1500 Z=0:NEXTI
1510 PRINT:PRINT"DATEIENDE !!":PRINT
1520 PRINT"WEITER MIT RETURN":PRINT
1530 INPUTX$
1540 RETURN
1550 IFO$="B"THEN1730
```

```
1560 IFD$="A"THEN1670
1570 OPEN1,4
1580 PRINT#1,D$(I,1);SPC(10-LEN(D$(I,1)));
1590 PRINT#1,D$(I,2);SPC(20-LEN(D$(I,2)));
1600 PRINT#1,D$(I,3);SPC(20-LEN(D$(I,3)));
1610 PRINT#1,D$(I,4)
1620 PRINT#1,SPC(3);D$(I,5);SPC(25-LEN(D$(I,5)));
1630 PRINT#1,D$(I,6);SPC(15-LEN(D$(I,6)));
1640 PRINT#1,D$(I,7)
1650 PRINT#1:CLOSE1
1660 RETURN
1670 OPEN2,4
1680 PRINT#2
1690 FORJ=1TO5:PRINT#2,D$(I,J):NEXT
1700 PRINT#2:PRINT#2:PRINT#2
1710 CLOSE2
1720 RETURN
1730 GOSUB2030:GOSUB2110
1740 PRINT:PRINT"WEITER (J)?"
1750 GETX$:IFX$<>"J"THEN1750
1760 RETURN
1770 REM **************
1780 REM DATEN LOESCHEN
1790 REM **************
1800 IFX>0THEN1820
1810 GOSUB 2230:RETURN
1820 GOSUB2030
1830 INPUT"NAME 1 : ";N1$
1840 INPUT"NAME 2 : ";N2$
1850 FORI=1TOX
1860 IFD$(I,2)=N1$ANDD$(I,3)=N2$THEN1900
1870 NEXTI
1880 PRINT"NAME NICHT GEFUNDEN!":PRINT
1890 PRINT"WEITER MIT RETURN":INPUTX$:RETURN
1900 GOSUB2030:GOSUB2110
1910 PRINT:PRINT"ADRESSE LOESCHEN (J/N)?"
1920 GETX$:IFX$<>"J"ANDX$<>"N"THEN1920
1930 IFX$="N"THENRETURN
1940 FORY=ITOX-1
1950 FORJ=1TO6
1960 D$(Y,J)=D$(Y+1,J)
1970 NEXTJ,Y
1980 FORJ=1TO6:D$(X,J)="":NEXTJ
1990 X=X-1
2000 PRINT"SATZ IST GELOESCHT!"
2010 PRINT"WEITER MIT RETURN!"
2020 INPUTX$:RETURN
2030 REM *************
2040 REM PROGRAMM-KOPF
2050 REM *************
2060 PRINTCHR$(147);
2070 PRINTTAB(8);"=========================="
2080 PRINTTAB(8);"A D R E S S E N D A T E I
2090 PRINTTAB(8);"==========================":PRINT:PRINT
2100 RETURN
2110 REM **************
```

```
2120 REM SATZAUSGABE
2130 REM ***************
2140 PRINT"ANREDE       : ";D$(I,1)
2150 PRINT"NAME 1       : ";D$(I,2)
2160 PRINT"NAME 2       : ";D$(I,3)
2170 PRINT"STRASSE/NR.  : ";D$(I,4)
2180 PRINT"PLZ/ORT      : ";D$(I,5)
2200 PRINT"TELEFON      : ";D$(I,6)
2210 PRINT"BEMERKUNG    : ";D$(I,7)
2220 RETURN
2230 REM *************
2240 REM KEINE DATEI!
2250 REM *************
2260 GOSUB2030
2270 PRINT"KEINE DATEI IM RECHNER!":PRINT
2280 PRINT"WEITER MIT RETURN"
2290 INPUTX$:RETURN
```

1.4.11 Anwendungsgebiete der sequentiellen Datenspeicheurng

Der große Vorteil der sequentiellen Datei gegenüber den in den nächsten Kapiteln beschriebenen relativen und Direktzugriffsdateien besteht vor allem im sehr sparsamen Umgang mit Speicherplatz. Daten der unterschiedlichsten Länge können fortlaufend hintereinander gespeichert werden, ohne daß Datensätze eine bestimmte definierte Länge haben müssen und jeweils nicht ausgenutzter Speicherplatz nutzlos vergeudet wird. Sinnvoll ausnutzen läßt sich dieser Vorteil überall dort, wo nicht ständig Teile der Datei geändert werden müssen, wo nicht laufend auf auf bestimmte Datensätze gezielt zugegriffen werden muß. Beispiele sind

* Protokolldateien
  In einem Buchungsjournal werden fortlaufend alle Buchungsvorgänge protokolliert. Änderungen sollen und d ü r f e n nicht vorgenommen werden.

* Auswertungsdateien
  Sie werten eine Direktzugriffsdatei aus, z.B. alle Kunden mit mehr als DM 5000,- Umsatz aus dem Postleitzahlengebiet 4, und schreiben die gefundenen Datensätze für den späteren Ausdruck in eine sequentielle Datei.

Natürlich bieten sich die sequentiellen Dateien auch, wie in den vorherigen Kapiteln beschrieben, als Ersatz für Direktzugriffsdateien an, wenn beim Anwender weiterreichende Programmierkenntnisse nicht vorhanden sind. Allerdings würden wir Ihnen empfehlen, auch die anderen Arten der Datenspeicherung durchzuarbeiten, da sie zum Teil gravierende Vorteile bieten.

1.5 Relative Datenspeicherung

Die relative Datenspeicherung und ihre Programmierung werden im Handbuch der VC-1541 nicht beschrieben. Der Grund dürfte darin liegen, daß der COMMODORE 64 und der VC-20 in ihrem BASIC 2.0 keine Befehle zur Verwaltung von relativen Dateien enthalten. Damit ist im Prinzip eine relative Datenspeicherung mit dem CBM 64 und dem VC-20 nicht möglich - aber nur im Prinzip. Wir haben einige Kunstgriffe entwickelt, mit denen Sie die Beschränkung des BASIC 2.0 umgehen und die relative Datenspeicherung auch mit dem VC-20 und dem 64-er nutzen können. Im einzelnen mag dies zwar manchmal etwas kompliziert erscheinen - so werden z.B. Angaben über die Recordlänge an die Floppy mit CHR$(x)-Codes übermittelt - ,doch erschließen Sie sich so eine sehr komfortable Methode der Datenspeicherung.

1.5.1 Das Prinzip

Bei der relativen Dateiverwaltung werden die Datensätze (auch Records genannt) durchnumeriert. Mit der Voraussetzung, daß alle Datensätze einer relativen Datei die gleiche Länge haben, kann anhand der Recordnummer jeder Datensatz direkt adressiert werden. Zum Auffinden eines Records ist es nicht erforderlich, die gesamte Datei zu durchsuchen. Es wird lediglich die Nummer des Records relativ zum Dateianfang angegeben und der Record kann ausgelesen werden. Anhand der Satznummer kann das DOS erkennen, wo sich der Datensatz "relativ" zum Anfang der Datei auf Diskette befindet und so direkt auf diesen Datensatz zugreifen. Damit müssen nicht mehr komplette Dateien oder Indextabellen in den Rechner eingelesen werden, sondern nur nuch die gerade benötigten Datensätze.

Die Verwaltung einer relativen Datei läuft nach folgendem Muster ab:

Einrichten einer relativen Datei:

1. Die Datei wird geöffnet. Dabei wird die Länge eines Records festgelegt.
2. Der letzte Record wird gekennzeichnet.
3. Die Datei wird wieder geschlossen.

Schreiben eines Records:

1. Die Datei wird geöffnet.
2. Es wird auf den zu schreibenden Record positioniert.
3. Der Record wird geschrieben.
4. Die Datei wird geschlossen.

Lesen eines Records:

1. Die Datei wird geöffnet.
2. Es wird auf den zu lesenden Record positioniert.
3. Der Record wird gelesen.
4. Die Datei wird geschlossen.

Dies war nur eine grobe Ubersicht. In den folgenden Abschnitten werden diese Vorgänge noch ausführlich beschrieben.

1.5.2 Der Vorteil gegenüber sequentieller Speicherung

Die wesentlichen Vorteile der relativen Speicherung sind:

* schneller Zugriff auf jeden Record

* relative Dateien entlasten den Speicher des Rechners

Bei der Behandlung der sequentiellen Dateien wurde bereits erwähnt, daß die sequentielle Datei zu deren Verwaltung vollständig im Speicher des Rechners enthalten sein muß. Ist dies nicht der Fall, so ist es beim Aufsuchen eines Datensatzes notwendig, die gesamte Datei zu durchsuchen. D.h. jeder Datensatz muß gelesen und mit dem Suchbegriff verglichen werden. Sollte eine sequentielle Datei nicht vollständig im Speicher unterzubringen sein, so ist diese Methode des Suchens unumgänglich.
Bei relativen Dateien ist das wesentlich einfacher. Mit Hilfe der Recordnummer kann auf jeden Satz direkt zugegriffen werden. Die Datei ist also vom Speicher des Rechners unabhängig. So kann z.B. mit einem Programm, das die 3,5 KByte des VC 20 vollständig belegt, eine Datei mit bis zu 163 KByte verwaltet werden!
Die Vorteile der relativen gegenüber der sequentiellen Dateiverwaltung sind derart groß, daß jeder, der einmal mit relativen Dateien vertraut ist, die Form der relativen Dateien vorziehen wird.

1.5.3 Das Öffnen einer relativen Datei

Auch relative Dateien werden mit einem OPEN-Befehl geöffnet. Dieser Befehl unterscheidet sich nur gering von dem der sequentiellen Dateien. Schauen Sie sich nun das Format des OPEN-Befehls einmal an:

```
OPEN lfn,ga,kanal,"filename,L,"+CHR$(recordlänge)
```

Die ersten 4 Parameter sind mit denen des OPEN-Befehls für sequentielle Dateien identisch. Also logische Filenummer, Geräteadresse (im Normalfall 8), Kanal (2-14), Name der

Datei.
Nun folgt ein 'L', das dem DOS mitteilt, daß nun eine relative Datei geöffnet werden soll, deren Recordlänge folgt. Diese Recorlänge wird mit einem CHR$-Code übermittelt. Die Länge liegt zwischen 1 und 254. Ein Record darf also maximal 254 Zeichen umfassen.
Ist die Recordlänge kleiner als 88, so kann der Record mit einem INPUT-Befehl gelesen werden. Dazu ist es aber erforderlich, daß der PRINT-Befehl den Record mit einem abschließenden RETURN übermittelt hat. In der Regel sendet der PRINT-Befehl dieses RETURN, wenn er nicht mit einem Semikolon abgeschlossen wurde. Dieses RETURN ist nun Bestandteil des Records. Wollen Sie also Records mit INPUT einlesen, so muß die Recordlänge im OPEN-Befehl immer um eins erhöht werden.
Eine Datei, deren 80-Zeichen umfassende Records mit INPUT eingelesen werden sollen würde demnach folgendermaßen geöffnet:

```
OPEN 1,8,2,"FILE.REL,L,"+CHR$(81)
```

Hier wird ein relatives File mit dem Namen "FILE.REL" über Kanal 2 geöffnet. Die Recordlänge soll 81 Zeichen betragen. Es sollen also 80 Zeichen umfassende Records mit einem PRINT-Befehl gesendet werden, dem kein Semikolon folgt.

Wichtig ist, daß immer nur eine relative Datei geöffnet sein kann. Wollen Sie mit zwei relativen Dateien arbeiten, so muß immer die erste geschlossen werden, bevor die zweite geöffnet wird. Zusätzlich zu der relativen Datei kann eine sequentielle Datei geöffnet werden.
Zum erstmaligen Einrichten einer relativen Datei ist es sinnvoll, den letzten Record freizugeben, da dann sämtliche vor diesem Record liegende Datensätze auch freigegeben werden. Freigeben bedeutet, den Record mit dem Byte CHR$(255) zu beschreiben. Versucht man, einen Reocrd zu lesen, dessen Nummer über die des letzten Records der Datei liegt, so verursacht dies den Fehler "RECORD NOT PRESENT". Beschreibt man jedoch einen Record, der über dem bisher höchsten Record liegt, so werden gleichzeitig alle Records, die unterhalb dieses neuen Records liegen, mit CHR$(255) beschrieben. Ein späterer Lesezugriff auf einen Record dieses Bereichs erfolgt dann fehlerlos. Das Beschreiben dieser "freigegebenen" Records erfolgt dann wesentlich schneller, weil alle Records, die unter diesem liegen, nicht mehr freigegeben werden müssen. Ein Beispiel:

Sie errichten eine relative Datei mit 100 Records. Sie geben aber den letzten (100.) Record nicht frei. Wenn Sie nun einen Record beschreiben, der über dem letzten beschriebenen Record dieser Datei angeordnet ist, werden gleichzeitig alle Records, die zwischen dem letzen und dem gerade beschriebenen Record liegen, freigegeben. Um diese Prozedur zu vermeiden, wird nach dem erstmaligem Öffnen der letzte Record, und somit auch alle anderen Records freigegeben. Das spätere Beschreiben dieses freigegebenen Records läuft dann

wesentlich schneller ab.
Zum Freigeben des letzten Records wird dieser also lediglich mit dem ASCII-Wert $FF -CHR$(255)- beschrieben. Zum Beschreiben eines Records muß aber vorher auf diesen positioniert werden. Dazu wird über dem Befehlskanal der Floppy (15) ein Positionier-Befehl gesendet werden, der wie folgt aufgebaut ist:

```
PRINT#1fn,"P"+CHR$(kanal)+CHR$(low)+CHR$(high)+CHR$(byte)
```

Wenn zum Freigeben von Records auf einem Record positioniert wird, der über das bisherige Dateiende hinausgeht, so erscheint im Floppy-Fehlerkanal die Meldung "RECORD NOT PRESENT". Da dieser positionierte Record aber nicht gelesen, sondern nur beschrieben (freigegeben) werden soll, kann die Meldung ignoriert werden. Der folgende PRINT auf diesen, noch nicht freigegebenen Record wird trotz der Fehlermeldung durchgeführt.

Die Parameter 'low' und 'high' im P-Befehl geben die Recordnummer an. Da mit einem Byte maximal der Wert 254 angegeben werden kann, eine relative Datei aber bis zu 65535 Records beinhaltet, muß die Recordnummer in zwei Bytes übermittelt werden. Diese zwei Bytes berechnet man mit folgender Formel:

```
HB=INT(RN/256)
LB=RN-HB*256
```

HB = High Byte (Parameter 'high')
LB = Low Byte (Parameter 'low')
RN = Recordnummer

Der letzte Parameter dient der Positionierung auf eine bestimmte Stelle innerhalb des angegebenen Records. Ein Beispiel:

```
PRINT#2,"P"+CHR$(2)+CHR$(10)+CHR$(1)+CHR$(5)
```

Hier wird auf das 5. Byte des 266. Records positioniert. Diese 266 wird als Lowbyte 10 und Highbyte 1 codiert (Highbyte * 256 + Lowbyte = Recordnummer)
Zum Lesen oder Schreiben eines kompletten Records muß unbedingt auf das 1. Byte positioniert werden. Wird der letzte Parameter nicht angegeben, so wird das abschließende 'RETURN' -CHR$(13)- als Bytepositionierung angenommen.

Der entsprechende BASIC-Ausschnitt zum Einrichten einer Datei mit 1000 Records und jeweils 80 Zeichen sieht dann folgendermaßen aus:

```
100 RN=1000
110 HB=INT(RN/256)
120 LB=RN-HB*256
130 OPEN1,8,2,"FILE.REL,L,"+CHR$(80)
140 OPEN2,8,15
```

```
150 PRINT#2,"P"+CHR$(2)+CHR$(LB)+CHR$(HB)+CHR$(1)
160 PRINT#1,CHR$(255)
170 CLOSE 1:CLOSE 2
```

Das Freigeben der 1000 Records nimmt einige Zeit in Anspruch. So kann das Einrichten dieser Datei ca. 10 Minuten dauern. Beachten Sie aber, daß in diesen 80-Zeichen-Records nur 79 Zeichen Daten untergebracht werden kann, wenn die Daten mit einem PRINT-Befehl mit abschließendem RETURN übertragen werden!

1.5.4 Vorbereitung der Daten zur relativen Speicherung

Wie bereits erwähnt, sind Sie bei der relativen Speicherung an eine feste Satzlänge gebunden. Besteht ein Record aus mehrerern Feldern, so müssen diese Felder zusammengefügt werden. Wichtig ist hierbei, daß sich die Felder in der gesamten Datei in jedem Datensatz immer an derselben Position befinden müssen. Spielen wir dieses Problem einmal durch:

Es soll ein Artikelstamm relativ verwaltet werden. Dazu sind folgende Felder notwendig:

ARTIKELNUMMER	4-stellig
BEZEICHNUNG	15-stellig
LAGERNUMMER	5-stellig
EINK.-PREIS	6-stellig
VERK.-PREIS	6-stellig
Recordlänge	36 Bytes

Der Artikelstamm umfasst ca. 200 Artikel mit einer Satzlänge von 36 Bytes. Diese Artikeldatei soll nun eingerichtet werden:

```
100 RN=200:REM ANZAHL DER ARTIKELSÄTZE
110 RL=36 :REM RECORDLÄNGE
120 OPEN 1,8,2,"ARTIKEL,L,"+CHR$(36)
130 OPEN 2,8,15
140 PRINT#2,"P"+CHR$(2)+CHR$(200)+CHR$(0)+CHR$(1)
150 PRINT#1,CHR$(255)
160 CLOSE 1:CLOSE 2
```

Nun ist die Datei eingerichtet und alle Records können beschrieben werden. Nehmen wir nun einmal an, daß die Artikeldatei sequentiell vorliegt. Sie besteht aus 200 Datensätzen deren Felder hintereinander angeordnet sind. Diese Felder müssen zu einzelnen Records zusammengebunden und in die relative Datei übertragen werden. Das ist aber nicht einfach, da z.B. die Artikelbezeichnung nicht immer die volle Länge von 15 Zeichen haben wird. Die Struktur der relativen Datei soll wie folgt aussehen:

Position	1111111111222222222233333333 12345678901234567890123456789012345678
Feld	AN$--BE$----------LN$--EP$---VP$---
Inhalt	1 BLECH 2MM 1344 23.40 42.30 2 SCHRAUBE 3MM 1231 9.00 14.00 3 VENTIL A3A4 1243 23.45 29.90 . . . 200 SCHLAUCH 12MM 2321 6.70 9.80

Aus der sequentiellen Datei werden die Felder in folgende Variablen eingelesen:

Artikelnummer	nach AN$
Artikelbezeichnung	nach BE$
Lagernummer	nach LN$
Einkaufspreis	nach EP$
Verkaufspreis	nach VP$

Der folgende Befehl verkettet zwar diese Felder, aber wie sich noch herausstellen wird, nicht mit dem erwünschten Erfolg:

```
RC$ = AN$ + BE$ + LN$ + EP$ + VP$
```

Dieser Record RC$ entspricht nicht der gewünschten Struktur der Datei. Der Grund dafür ist, daß der Artikelbezeichnug dann unmittelbar die Lagernummer folgt. Da die Lagernummer aber unbedingt ab Stelle 20 beginnen muß und die Artikelbezeichnug nicht konstant 15 Zeichen umfasst, ergeben sich dabei Probleme. Um die Records nach dem Lesen aus der relativen Datei wieder richtig aufbereiten zu können, muß die Struktur unbedingt eingehalten werden. Dazu müssen alle Felder, falls sie kürzer als die eingeplante Länge sind, mit Leerzeichen aufgefüllt werden. Wenn man dies berücksichtigt, sieht die Verkettung folgendermaßen aus:

```
BL$="                "
RC$=AN$+LEFT$(BL$,4-LEN(AN$))
RC$=RC$+BE$+LEFT$(BL$,15-LEN(BE$))
RC$=RC$+LN$+LEFT$(BL$,5-LEN(LN$))
RC$=RC$+EP$+LEFT$(BL$,6-LEN(EP$))
RC$=RC$+VP$+LEFT$(BL$,6-LEN(EP$))
```

Diese Verkettung sieht komplizierter aus, als sie es wirklich ist. Jedes Feld muß mit der Anzahl von Leerzeichen ergänzt werden, die sich aus max. Länge des Feldes minus tatsächlicher Länge des Feldes ergibt. Diese Leerzeichen werden dem anfangs definierten String BL$ entnommen. Dieser String ist so lang wie das längste Feld des Records, in diesem Fall 15 Zeichen.

Spielen wir einmal ein Beispiel durch: Angenommen die erste Artikelnummer ist 8. Die Länge dieses Strings ,LEN(AN$), ist also 1. Die max. Länge des Feldes (4) minus der tatsächlichen Länge (1) ergibt also 3. Der String AN$ muß also mit 3 Leerzeichen ,LEFT$(BL$,3), aufgefüllt werden.
Jeder Datensatz der bisherigen, sequentiellen Datei muß derartig aufbereitet werden, bevor man ihn in die relative Datei übernehmen kann.
Natürlich gilt das oben gesagte für alle Eingabewerte, die in die relative Datei übernommen werden sollen. Denken Sie deshalb bei der Programmierung der relativen Dateiverwaltung immer an die Benutzung einer Routine zum Auffüllen der einzelnen Felder bis zur Sollänge mit Leerzeichen.

1.5.5 Datenübertragung Floppy / Rechner

Im Prinzip unterscheidet sich die Datenübertragung nicht von der bei der sequentiellen Speicherung. Sätze werden mit PRINT geschrieben und mit INPUT bzw. GET wieder gelesen. Der einzige Unterschied ist, daß vor Lesen oder Schreiben eines Records auf diesen positioniert werden muß. Dies geschieht mit dem P-Befehl. Erstellen wir nun einmal mit folgendem Programm eine relative Datei im Dialog:

```
100 BL$="                    "
105 OPEN 1,8,2,"TEST.REL,L,"+CHR$(41)
110 OPEN 2,8,15
120 PRINT#2,"P"+CHR$(2)+CHR$(100)+CHR$(0)+CHR$(1)
130 PRINT#1,CHR$(255)
140 PRINT CHR$(147)
150 PRINT"DATENSATZEINGABE:"
160 PRINT"----------------"
170 INPUT"RECORDNUMMER (1-100)    : ";RN
180 IF RN<1 OR RN>100 THEN PRINT CHR$(145);:GOTO160
190 INPUT"FELD 1 (MAX.10 ZEICHEN): ";F1$
200 IF LEN(F1$)>10 THEN PRINT CHR$(145);:GOTO190
210 INPUT"FELD 2 (MAX. 5 ZEICHEN): ";F2$
220 IF LEN(F2$)>5 THEN PRINT CHR$(145);:GOTO210
230 INPUT"FELD 3 (MAX.10 ZEICHEN): ";F3$
240 IF LEN(F3$)>10 THEN PRINT CHR$(145);:GOTO230
250 INPUT"FELD 4 (MAX.15 ZEICHEN): ";F4$
260 IF LEN(F4$)>15 THEN PRINT CHR$(145);:GOTO250
270 PRINT"RICHTIG (J/N)?"
280 GET X$:IF X$<>"J" AND X$<>"N"THEN280
290 IF X$="N"THEN 140
300 RC$=F1$+LEFT$(BL$,10-LEN(F1$))
310 RC$=RC$+F2$+LEFT$(BL$,5-LEN(F2$))
320 RC$=RC$+F3$+LEFT$(BL$,10-LEN(F3$))
330 RC$=RC$+F4$+LEFT$(BL$,15-LEN(F4$))
340 PRINT#2,"P"+CHR$(2)+CHR$(RN)+CHR$(0)+CHR$(1)
350 PRINT#1,RC$
360 PRINT"WEITERE EINGABEN (J/N)?"
370 GET X$:IF X$<>"J"AND X$<>"N"THEN 370
380 IF X$="J"THEN 140
```

```
390 CLOSE 1:CLOSE 2:END
```

Die folgende, zeilenorientierte Dokumentation verdeutlicht die Arbeitsweise dieses Programms:

100	Es wird ein Leerzeichen-String mit der Länge 15 definiert.
105	Die relative Datei mit der Länge 41 wird geöffnet.
110	Der Befehlskanal 15 wird geöffnet.
120	Zum Initialisieren der relativen Datei wird auf das 1. Byte des letzten (100.) Satzes positioniert.
130	Der letzte Satz wird freigegeben und die Initialisierung beginnt.
140	Der Bildschirm wird gelöscht.
150-260	Die Recordnummer und die Felder 1-4 werden eingegeben und auf korrekte Länge geprüft.
270-290	Die eingegebenen Daten können noch einmal korrigiert werden.
300-330	Der Record wird aufbereitet.
340	Es wird auf das 1. Byte des angegebenen Records positioniert.
350	Der Record wird auf Diskette geschrieben.
360-380	Es können erneut Daten eingegeben werden.
390	Das Programm wird beendet

Erfassen Sie nun mit diesem Programm einige Records. Vergessen Sie aber nicht, dieses Programm abzuspeichern, falls Sie es später noch benötigen.

Sicherlich ist es auch notwendig, erfasste Daten zu lesen und verändern. Dazu wird die relative Datei geöffnet, auf den gewünschten Record positioniert und eingelesen. Dieser Record muß dann wieder in seinen Feldern zerlegt werden. Lesen wir nun einmal gezielt einen Record, der mit der o.g. Routine erfasst wurde. Die folgende Routine liest diesen Record:

```
100 OPEN 1,8,2,"TEST.REL,L,"+CHR$(41)
110 OPEN 2,8,15
115 PRINT CHR$(147)
120 INPUT"RECORNUMMER :";RN
130 PRINT#2,"P"+CHR$(2)+CHR$(RN)+CHR$(0)+CHR$(1)
140 INPUT#1,RC$
160 IF ASC(RC$)=255 THEN PRINT"RECORD NICHT
    BELEGT!":GOTO 250
170 PRINT RC$
250 CLOSE 1:CLOSE 2
```

Diese Routine liest einen bestimmten Record. Ist dieser Record nicht belegt, so wird dies an den Wert 255 erkannt, mit dem beim Einrichten der Datei jeder freie Record gekennzeichnet wird.
Ein beschriebener Record wird angezeigt. Sie erkennen dabei, daß die Felder 1-4 immer an derselben Stelle enthalten sind. Wollen Sie den Record wieder in seine einzelnen Felder aufteilen, so müssen diese mit dem Befehl MID$ dem Record entnommen werden. Um z.B. das Feld 1 dem Record zu entnehmen,

geben Sie nach Auffinden eines Records im Direkt-Modus folgende Befehle ein:

```
F1$=MID$(RC$,1,10)
PRINT F1$
```

Nun befindet sich in der Variablen F1$ das 1. Feld, wie es im Erfassungsprogramm eingegeben wurde. Dieses "Zerpflücken" des Records können Sie in die o.g. Routine einbauen. Geben Sie dazu folgende Zeile zusätzlich ein:

```
170 F1$=MID$(RC$,1,10)
180 F2$=MID$(RC$,11,5)
190 F3$=MID$(RC$,16,10)
200 F4$=MID$(RC$,26,15)
210 PRINT"FELD 1: ";F1$
220 PRINT"FELD 2: ";F2$
230 PRINT"FELD 3: ";F3$
240 PRINT"FELD 4: ";F4$
250 PRINT"WEITERER ZUGRIFF (J/N)?"
260 GETX$:IF X$<>"J"AND X$<>"N"THEN 260
270 IF X$="J"THEN 115
280 CLOSE 1:CLOSE 2
```

Hier wird der Record aufbereitet und die Felder angezeigt. Wichtig hierbei ist, daß die Angaben im MID$-Befehl der genauen Position des Feldes innerhalb des Records entsprechen müssen. Die erste Angabe innerhalb der Klammer ist die Stringvariable, aus der ein Ausschnitt entnommen werden soll. Die zweite Angabe ist die Position, ab der die Anzahl Zeichen entnommen werden soll, die in der dritten Angabe bestimmt ist.
Mit den selektierten Feldern kann nun innerhalb des Programms weiter gearbeitet werden.
Bisher haben wir die Records mit dem INPUT-Befehl eingelesen. Ist der Record aber länger als 88 Zeichen, so kann er mit dem INPUT-Befehl nicht mehr eingelesen werden. Der Grund dafür ist, das ein INPUT-Befehl grundsätzlich nicht mehr als 88 Zeichen einlesen kann. Die Ausweichmöglichkeit zu dem nur beschränkt einsetzbaren INPUT-Befehl ist der GET-Befehl. Mit diesem Befehl werden die Bytes des Records einzeln gelesen und zu einem String verkettet. Nehmen wir einmal an, Sie haben eine relative Datei mit 128 Zeichen eingerichtet und diese auch beschrieben. Nun wollen Sie den 10. Record dieser Datei lesen und in die Variable RC$ übernehmen. Das Beispiel der folgenden Routine verdeutlicht dieses Einlesen mit GET:

```
100 OPEN 1,8,2,"TEST.GET,L,"+CHR$(128)
110 OPEN 2,8,15
120 PRINT#2,"P"+CHR$(2)+CHR$(10)+CHR$(0)+CHR$(1)
130 RC$=""
140 FOR I=1 TO 128
150 GET#1,X$
160 RC$=RC$+X$
170 NEXT I
```

Nach Ablauf dieser Routine steht der Record in der Variablen RC$ zur Verfügung. Ist dieser Record mit einem PRINT-Befehl ohne anschließendes Semikolon übertragen worden, das ein RETURN unterdrückt, so ist das letzte Zeichen in dem String RC$ ein RETURN. Um dieses RETURN zu ignorieren, läßt man die Schleife in Zeile 140 nur bis 127 laufen. das letzte Zeichen des Records (das RETURN) wird nun nicht gelesen.
Wie bereits erwähnt, gibt der letzte Parameter des P-Befehls an, ab welchem Zeichen des Records gelesen werden soll. Wenn Sie z.B in dem 127-Zeichen-Record des vorherigen Beispiels ein an der Position 40-60 befindliches Feld lesen möchten, so wird auf das 40. Zeichen positioniert und die folgenden 21 Zeichen eingelesen. Die folgende Routine verdeutlicht dies:

```
100 OPEN 1,8,2,"TEST.GET,L,"+CHR$(128)
110 OPEN 2,8,15
120 PRINT#2,"P"+CHR$(2)+CHR$(10)+CHR$(0)+CHR$(40)
130 F$=""
140 FOR I= 1 TO 21
150 GET#1,X$
160 F$=F$+X$
170 NEXT I
    .
    .
    .
    .
```

Da in der Zeile 120 auf das 40. Byte des 10. Records positioniert wird und die Schleife in den Zeilen 140-170 die folgenden 21 Bytes (Bytes 40-60 des Records) in F$ einliest, befindet sich das dort enthaltene Feld nach Ablauf dieser Routine in F$.
Sie sehen also, daß zum Arbeiten mit einem Teil des Records nicht der gesamte Record eingelesen werden muß. Der Positionier-Befehl ermöglicht dies.

1.5.6 Schließen einer relativen Datei

Beim Schließen einer relativen Datei gibt es keine Unterschiede zur sequentiellen Speicherung. Da aber zur Verwaltung einer relativen Datei immer der Befehlskanal 15 zum Senden des Positionierbefehls offen gehalten werden muß, muß auch dieser geschlossen werden. Selbstverständlich muß die Filenummer, die beim OPEN-Befehl gewählt wurde, auch beim Schließen dem File bzw. dem Befehlskanal entsprechen.

1.5.7 Suchen eines Records nach der binären Methode

Im Normalfall wird auf jedem Record mit der Recordnummer zugegriffen. Nun kann es aber z.B. vorkommen, daß in einer relativen Adressendatei der Herr Müller gesucht wird, die entsprechende Recordnummer aber nicht bekannt ist. Nun muß der Herr Müller gesucht werden. Eine Möglichkeit ist, jeden Record zu lesen, mit dem Namen Müller zu vergleichen, usw. Das kann bei einer Datei, die vieleicht 1000 Adressen enthält, sehr zeitaufwendig sein. Liegt die Datei in sortierter Form vor, so kann dieser Record mit einer anderen Methode gesucht werden. Diese Methode nennt man "binäres Suchen". Hierbei ist es aber unbedingt notwendig, die Datei sortiert aufrechtzuhalten. Wird z.B. ein Record hinzugefügt, so muß dieser entsprechend eingeordnet werden.

Das binäre Suchen kann man an einem einfachen Beispiel verdeutlichen: Wenn Sie z.B. in einem Telefonbuch nach einer Telefonnummer suchen, so gehen Sie sicher nicht sequentiell vor. Sie schlagen die Mitte des Buches auf und vergleichen, ob der erste Buchstabe des gesuchten Namens dem aufgeschlagenen Teil entspricht. Ist der gesuchte Name kleiner, so schlagen Sie die Hälfte des 1. Teils auf, usw. Sie gehen also systematisch vor.
Beim binären Suchen wird nicht sequentiell weitergesucht, wenn ein Record gefunden ist, der nicht dem gesuchten Record entspricht. Es wird anschließend auf den Record zugegriffen, der jeweils durch Zweiteilung der restlichen Anzahl der Datensätze ermittlelt wird. Daa folgende Beispiel verdeutlicht dies:

Es existiert folgende, aufwärts sortierte, relative Datei:

Recordnummer	Inhalt
1	1985
2	1999
3	2005
4	2230
5	2465
6	2897
7	3490
8	3539
9	4123
10	5000
11	5210
12	6450
13	6500
14	6550
15	6999

Von diesen aufgeführten 15 Records wird der Record mit dem Inhalt 3490 gesucht. Es ist nicht bekannt, auf welchem Platz er gespeichert ist.
Zunächst ist festzustellen, aus wieviel Records die Datei besteht. Im vorliegenden Fall aus 15. Die festgestellte Zahl ist durch zwei zu teilen. Diese Mitte der Datei stellt den 8. Record mit dem Inhalt 3539 dar. Es ist nun festzustellen, ob dieser Record den Suchbegriff 3490 enthält und falls nicht,

ob der Suchbegriff größer oder kleiner als der vorgefundene Inhalt, in diesem Fall 3539 ist. Das Vergleichsergebnis zeigt kleiner an. Somit befindet sich der gesuchte Record in der Menge der Records, die kleiner als der Vorgefundene ist. Es ist also auf die Mitte dieser Restes zuzugreifen. Wir erhalten den Record 4 mit dem Inhalt 2230. Das Vergleichsergebnis zeigt an, daß der Suchbegriff 3490 größer als der bei Record 4 vorgefundene Inhalt 2230 ist. Der dritte Zugriff geht auf die Mitte zwischen dem 4. und dem 8. Record, also auf den 6. Record mit dem Inhalt 2897. Das Vergleichsergebnis zeigt abermals kleiner an; das bedeutet, es ist die Mitte zwischen den Records 6 und 8 zu bilden. Somit ist der Suchbegriff unter Record 7 zu finden.
Das Prinzip des binären Suchens besteht darin, das jeweils, je nach Vergleichsergebnis, aufwärts oder abwärts die Mitte zu suchen ist, bis der Suchbegriff gefunden wurde. Die maximale Anzahl der Suchvorgänge errechnet sich nach folgender Formel:

```
S=INT(LOG(N)/LOG(2)+1)
```

Hierbei ist S die Anzahl der Zugriffe und N die Anzahl der Records der Datei. In einer sortierten, relativen Datei mit z.B. 1000 Records werden maximal 10 Zugriffe zum Aufsuchen eines beliebigen Records benötigt!
Erstellen wir die relative Datei mit 15 Datensätze, um anschließend innerhalb dieser Datei binär zu suchen:

```
100 OPEN1,8,2,"BINAER.REL,L,"+CHR$(5)
110 FORI=1TO15
120 READ RC$
130 PRINT#1,RC$
140 NEXT I
150 CLOSE 1:CLOSE 2:END
160 DATA 1985,1999,2005,2230,2465,2897,3490,3539
170 DATA 4123,5000,5210,6450,6500,6550,6999
```

Dieses Programm erstellt die 15 Records umfassende Datei "BINAER.REL" mit den in Zeile 160 bis 170 angegebenen Werten. Hier wird der Positionier-Befehl nicht benötigt, da die Datei vom ersten bis zum letzten Satz komplett beschrieben wird. Der Zeiger steht also nach Eröffnen der relativen Datei auf dem ersten Record. In dieser Datei sollen nun Records binär gesucht werden. Das folgende Programm ist nach der Logik des binären Suchens aufgebaut:

```
100 OPEN1,8,2,"BINAER.REL,L,"+CHR$(5)
110 OPEN2,8,15
120 PRINTCHR$(147)
140 N=15: REM ANZAHL RECORDS
150 I=LOG(N)/LOG(2)
160 IFI-INT(I)<>0THENI=INT(I)+1
170 M=I-1
180 I=2↑I
190 X=I/2
210 INPUT"SUCHBEGRIFF (* FUER ENDE): ";SB$
```

```
220 IFSB$="*"THEN 320
230 IF M<0 THEN PRINT"RECORD NICHT GEFUNNDEN":
    GOTO140
240 M=M-1
250 PRINT#2,"P"+CHR$(2)+CHR$(X)+CHR$(0)+CHR$(1)
260 INPUT#1,RC$
270 IF SB$=RC$ THEN 340
280 IF SB$<RC$ THEN X=X-2↑M:GOTO230
290 X=X+2↑M
300 IF X>I THEN PRINT"DATEI UEBERSCHRITTEN":GOTO140
310 GOTO 230
320 CLOSE 1:CLOSE 2
330 END
340 PRINT"RECORD GEFUNDEN! "
350 PRINT"INHALT :"RC$
360 GOTO140
```

Die Dokumentation des Programms:

100	Die relative Datei "BINAER.REL" wird geöffnet.
110	Der Befehlskanal wird geöffnet.
120	Der Bildschirm wird gelöscht.
140	Die Anzahl der Records wird in der Variablen N gespeichert.
150-190	Sofern die maximale Anzahl der Records keine Zweierpotenz darstellt, wird die nächsthöherer Zweierpotenz gebildet. Dabei wird der Dateibereich zwar nach oben erweitert, aber es gehen auch keine Records verloren. Der Exponent dieser Zweierpotenz wird als Index benutzt. X wird der Wert I/2 zugeordnet. I/2 bezeichnet die genau die Mitte der (erweiterten) Datei. Außerdem wird wird die Variable M angelegt, die den Anfangswert I-1 enthält.
210-220	Der Suchbegriff wird eingelesen. Soll das Programm beendet werden, so wird '*' eingegeben.
230	Wenn M<0 , dann ist der Suchbegriff nicht gefunden worden.
240	M wird um eins vermindert. Die nächste Potenzierung mit M ergibt also die Hälfte des Restes der Datei.
250-260	Es wird auf den Record positioniert, dessen Nummer in der Variablen X enthalten ist.
270	Entspricht der eingelesene Record dem Suchbegriff, dann wird die Suche abgebrochen und der Record ausgegeben.
280-310	Es wird festgestellt, ob der Suchbegriff kleiner oder größer als der gelesene Record ist. Dementsprechend wird die Mitte des oberen oder unteren Restes in die Variable X gespeichert und erneut eingelesen.
320-330	Die Dateien werden geschlossen und das Programm beendet.
340-360	Der gefundene Record wird ausgegeben.

Dieses, in BASIC codierte, binäre Suchen ist universell einsetzbar. Es müssen nur die Anzahl der Records und die Vergleiche Suchbegriff/Record entsprechend angepasst werden.

Benutzen Sie also diese Suchroutine zum Auffinden von Records in Ihren sortierten, relativen Dateien.

1.5.8 Suchen eines Records über separate Index-Dateien

Wenn Sie häufig auf einzelne Datensätze gezielt und schnell mit alphanumerischen Schlüsseln zugreifen wollen, die nicht der logischen Satznummer entsprechen, und Sie Ihre Datei nicht in entsprechend sortierter Form halten wollen, so empfiehlt sich eine andere Methode.
Bilden Sie für jeden gewünschten Schlüssel-Begriff bzw. Index eine eigene Index-Datei, in der pro Datensatz abgelegt sind

- jeweiliger Index
- zugehörige Satznummer

Diese Datei laden Sie bei Bedarf und zur Pflege ganz in den Speicher. Ein Beispiel:

Die haben als relative Datei Ihre Adressverwaltung angelegt, bestehend aus

- Vorname
- Name
- Straße
- PLZ
- Wohnort
- Telefonnummer

Sowohl nach dem Vornamen, als auch nach dem Namen möchten Sie gezielt suchen können. Also bilden Sie zwei zusätzliche relative Dateien, die als Felder nur den gewünschten Schlüsselbgriff, z.B. den Vornamen, und die Satznummer des entsprechenden Datensatzes in der Hauptdatei enthalten.

Die gewünschten Indexdateien sollten Sie jeweils komplett im Speicher halten, da dort schnellstmögliche Indexsuche erfolgen kann. Wollen Sie z.B. auf den Datensatz zugreifen, der als Vornamen "OTTO" hat, so durchsuchen Sie im Speicher die entsprechende Indexdatei und greifen anschließend mit der gefundenen Satznummer direkt auf den gewünschten Satz Ihrer Adressdatei zu.
Verfolgen Sie nun ein Beispiel:

Wir nehmen an, es existiert eine Hauptdatei und eine Indexdatei für den Namen:

Hauptdatei:			Indexdatei:		
Name	Vorname	weitere Felder	Index (Name)	Satznr. LB	HB
Walter	Karl	.....	Walter	01	00
Berger	Rainer	.....	Berger	02	00

```
Tietz       Klaus      .............   Tietz.          03  00
Schacht     Rolf       .............   Schacht         04  00
.           .                          .               .
.           .                          .               .
.           .                          .               .
Horstner    Gustav     .............   Horstner        99  00
-------------------------------------------------------------
```

Die Datei beinhaltet also 99 Datensätze. Bevor mit dem Programm gearbeitet werden kann, muß die Indextabelle eingelesen werden. Dies kann z.B. eine sequentielle Datei sein, die in der mit DIM IT$(99) reservierten Speichertabelle eingelesen wird. Die ersten 20 Zeichen eines jeden Indextabellen-Platzes stellt den Vornamen dar. Das vorletzte Byte (Nr.21) ist das Lowbyte und das letzte Byte (Nr.22) das Highbyte der Satznummer. Unter dieses Voraussetzungen kann mit folgender Routine ein beliebiger Datensatz aufgesucht werden:

```
            .
            .
            .
      100 INPUT "NAME";N$
      110 FOR I=1TO99
      120 IF LEFT$(IT$,20)=N$THEN 150
      130 NEXT I
      140 PRINT "NAME NICHT GEFUNDEN!":END
      150 PRINT"DATENSATZ GEFUNDEN!"
      160 OPEN1,8,2,"ADRESSEN,L,"+CHR$(81)
      170 OPEN 2,8,15
      180 PRINT#2,"P"+CHR$(2)+MID$(IT$,21,1)+CHR$(0)
          +CHR$(1)
      190 INPUT#1,RC$
            .
            .
            .
```

Die Schleife in Zeile 110-130 durchläuft sequentiell die Indextabelle nach dem gesuchten Namen, der sich in den linken 20 Zeichen befindet. Wird der Name nicht gefunden, so wird die Schleife verlassen und in Zeile 140 eine entsprechende Meldung ausgegeben, bevor das Programm beendet wird.
Wird in Zeile 120 eine Übereinstimmung zwischen Index und gesuchtem Namen festgestellt, so wird nach Zeile 150 verzweigt. Nach Ausgabe der Meldung wird die Adressendatei geöffnet (falls sie nicht vorher geöffnet wurde). NAch dem Öffnen des Befehlskanals wird der Positionierbefehl zur Floppy gesendet. Da im vorletzten Byte eines Indexeintrages das Lowbyte der Satznummer enthalten ist, braucht dies lediglch als MID$-Befehl eingebaut werden. Das Highbyte ist bekanntlich null, wenn die Satznummer 255 nicht überschreitet.
In Zeile 190 wird der Datensatz dann eingelesen und steht zur Verfügung.

Der Zugriff über Indexdateien stellt ebenfalls eine sehr

schnelle und ungemein flexible Form der Dateiorganisation dar. Theoretisch können Sie beliebig viele Indexdateien pro Hauptdatei anlegen. Allerdings müssen Sie zwei wichtige Einschränkungen beachten:

1. Bei Änderungen in der Hauptdatei, die Schlüsselfelder betreffen, müssen auch die entsprechenden Indexdateien gepflegt werden. Dies kann, besonders bei mehreren Indexdateien, sehr aufwendig sein.

2. Zahl und Größe der Indexdateien, die Sie zum Zweck des schnellen Zugriffs im Speicher Ihres Computers halten, werden durch den verfügbaren Speicherplatz begrenzt.

1.5.9 Ändern von Records

Der logische Ablauf zum Ändern eines Records ist folgender:

1. Einlesen des Records
2. "Splitten" des Records in seine Felder
3. Ändern der entsprechenden Felder
4. Zusammenfügen der Felder zu einem Record
5. Zurückschreiben des Records

Im Abschnitt 1.5.5 haben Sie einige Records in die Datei "TEST.REL" geschrieben. Diese Datei hat folgende Eigenschaften:

```
Recordlänge:              41 Bytes
Anzahl Records:          100
Anzahl Felder:             4
Länge, Position Feld 1: 10,  1-10
"    , "        Feld 2:  5, 11-15
"    , "        Feld 3: 10, 16-15
"    , "        Feld 4: 15, 26-40
abschließendes RETURN?: ja, Position 41
```

Eine derartige Dateibeschreibung sollten Sie für jede Ihrer Dateien anlegen. Dies ist z.B. sehr wichtig, wenn andere Programme auf diese Daten zugreifen sollen.
In dieser Datei sollen nun Records geändert werden. Das folgende Programm erfüllt diese Aufgabe:

```
100 REM ========================
110 REM      VORBEREITUNGEN
120 REM ========================
130 BL$="                      "
140 OPEN 1,8,2,"TEST.REL,L,"+CHR$(41)
150 OPEN 2,8,15
160 REM ========================
170 REM      RECORD EINLESEN
180 REM ========================
190 PRINT CHR$(147)
200 INPUT"RECORDNUMMER (1-100): ";RN
```

```
205 IF RN<1 OR RN>100 THEN PRINT CHR$(145);:GOTO200
210 PRINT"--------------------------"
220 PRINT#2,"P"+CHR$(2)+CHR$(RN)+CHR$(0)+CHR$(1)
230 INPUT#1,RC$
240 IF ASC(RC$)<>255 THEN 270
250 PRINT "RECORD UNBESCHRIEBEN"
260 GOTO 630
270 REM ========================
280 REM   RECORD AUFBEREITEN
290 REM ========================
300 F#(1)=MID$(RC$,1,10)
310 F$(2)=MID$(RC$,11,5)
320 F$(3)=MID$(RC$,16,10)
330 F$(4)=MID$(RC$,26,15)
340 REM ========================
350 REM     FELDER ANZEIGEN
360 REM ========================
370 PRINT CHR$(147)
380 FOR I=1 TO 4
390 PRINT"FELD";I;": ";F$(I)
400 NEXT I
410 PRINT"--------------------------"
420 REM ========================
430 REM      FELDER ÄNDERN
440 REM ========================
450 PRINT"WELCHES FELD SOLL GEÄNDERT WERDEN (1-4)?"
460 GET X$:IF X$<"1" OR X$>"4" THEN 460
470 INPUT"NEUER INHALT : ";F$(VAL(X$))
480 PRINT"RECORD IST GEÄNDERT"
490 PRINT"NOCH ÄNDERUNGEN IN DIESEM RECORD (J/N)?"
500 GET X$:IF X$<>"J" AND X$<>"N" THEN 500
510 IF X$="J"THEN 340
520 REM ========================
530 REM     FELDER VERKETTEN
540 REM ========================
550 RC$=F$(1)+LEFT$(BL$,10-LEN(F$(1)))
560 RC$=RC$+F$(2)+LEFT$(BL$,5-LEN(F$(2)))
570 RC$=RC$+F$(3)+LEFT$(BL$,10-LEN(F$(3)))
580 RC$=RC$+F$(4)+LEFT$(BL$,15-LEN(F$(4)))
590 REM ========================
600 REM RECORD ZURÜCKSCHREIBEN
610 REM ========================
620 PRINT#1,RC$
630 REM ========================
640 REM      PROGRAMM ENDE?
650 REM ========================
660 PRINT"NOCH ÄNDERUNGEN IN DER DATEI (J/N)?"
670 GET X$:IF X$<>"J" AND X$<>"N" THEN 670
680 IF X$="J" THEN 160
690 CLOSE 1: CLOSE 2: END
```

Nachdem Sie dieses Programm eingegeben und gestartet haben, können Sie nun beliebige Records ändern. Diese Records müssen allerdings mit dem im Abschnitt 1.5.5 enthaltenen Programm erfasst worden sein!
Dieses Änderungsprogramm prüft die neuen Feldeingaben nicht

auf korrekte Länge, sondern schneidet die Überlänge ab.
Die wesentlichen Befehlsfolgen in diesem Programm sind in den entsprechenden Abschnitten bereits ausführlich beschrieben worden.

1.5.10 Ergänzen einer relativen Datei

Jede relative Datei hat eine vom Anwender festgelegte Recordzahl. Diese kann beim Einrichten der Datei festgelegt werden, indem der letzte Record mit dem Wert CHR$(255) beschrieben wird. Dieses Beschreiben des letzten Records hat zu Folge, daß jeder Record unterhalb dieser Höchstgrenze ebenfalls mit CHR$(255) beschrieben, also zum Beschreiben freigegeben wird.
Die zweite Möglichkeit ist, daß die Datei beim Einrichten nicht in Ihrem vollen Umfang freigegeben wird. Wird z.B. der Record mit der Nummer 3 in die neue Datei geschrieben, so wird gleichzeitig der Record 1 und 2 freigegeben, also mit CHR$(255) beschrieben. Ein weiteres Beschreiben des 90. Records hat dann zur Folge, daß gleichzeitig die Records 4 bis 89 freigegeben werden, usw. Legen Sie also beim Einrichten der Datei nicht den letzten Record fest, so ist beim anschließendem Beschreiben der Datei mit einer wesentlich längeren Verarbeitungszeit zu rechnen. Es ist also sinnvoll, die Datei zu Anfang in Ihrem vollen Umfang freizugeben.
Eine eingerichtete, relative Datei kann jederzeit, sofern es die Diskettenkapazität erlaubt, vergrößert werden. Dazu wird der neu ermittelte, letzte Record mit CHR$(255) beschrieben. Gleichzeitig werden dann alle Records zwischen dem alten und dem neuen Dateiende freigegeben.
Ein Schreibzugriff auf eine relative Datei, der über das Dateiende hinausgeht, hat also keinen Fehler zur Folge. Wenn die Diskette diese Erweiterung ermöglicht, wird die Datei lediglich vergrößert. Ist ein Erweitern der Datei aufgrund mangelner Diskettenkapazität nicht möglich, so wird der Fehler "FILE TO LARGE" dem Fehlerkanal der Floppy übergeben.
Ein das Dateiende überschreitender Lesezugriff jedoch verursacht den Fehler "RECORD NOT PRESENT" im Floppy-Fehlerkanal.

1.5.11 "HAUSHALTSBUCH" mit relativer Datenspeicherung

Ein Beispiel einer kompletten Problemlösung mit relativer Datenspeicherung bietet Ihnen einen guten Einblick in die Organisation von relativen Dateien. Es soll Ihnen verdeutlichen, wie man die Idee zu einem Programm realisiert. Gleichzeitig ist dieses Programm für fast alle Besitzer dieses Buches einsetzbar .

Die Idee, ein Computer-Haushaltsbuch zu führen ist nicht gerade neu. Viele Programmierer haben sich mit diesem Problem

beschäftigt. Doch die wenigsten lösten dieses mit Hilfe der relativen Speicherung. Da die einzelnen Konten eines Haushaltsbuches numeriert werden, eignen sich diese Nummern sehr gut als Schlüssel zu dem entsprechenden Record. Die Kontonummer stellt also gleichzeitig die Recordnummer dar.
Die nächste Überlegung war, wie ein Record eines Kontos aufgebaut sein muß. Um die Konten nicht nur mit Nummern, sondern auch mit Klartext-Bezeichnung zu versehen, ist das erste Feld des Records der Kontenname. Wir haben diesen Namen auf 20 Zeichen festgelegt.
Da für jedes Jahr eine Datei geführt werden soll, sind 12 Felder notwendig, um die Kontensummen im Record unterzubringen. Diese Summenfelder sind jeweils 10 Zeichen groß. Diese Kontensummen werden als Strings abgelegt, die mit Hilfe des VAL-Befehls in numerische Variablen umgesetzt werden, um sie zu aktualisieren. Der Record umfaßt somit 141 Zeichen (20 für Name, 12*10 für Monatssummen und 1 für RETURN)

Der Aufbau des Records:

Feld	Länge	Position
Kontenname	20	1-20
Summe Januar	10	21-30
Summe Februar	10	31-40
.		
.		
.		
.		
Summe November	10	121-130
Summe Dezember	10	131-140

Wir haben die maximale Anzahl der Konten auf 20 begrenzt. Somit umfaßt eine Jahresdatei 20 Records mit je 141 Bytes. Diese Dateistruktur war Grundlage jeder weiteren Überlegungen.
Die nächste Überlegung war, welche Funktionen dieses Programm bieten sollte. Dabei legten wir uns auf folgende Programmteile fest:

* Konten anlegen
* Buchen
* Kontenübersicht
* Kontennamen ausgeben
* Monatsübersicht
* Jahresübersicht

Konten anlegen:

Dieses Unterprogramm errichtet die Datei für ein Jahr. Es wird die Anzahl der Konten und deren Namen abgefragt. Die jeweiligen Records werden dann mit den Kontennamen und den auf 0 gesetzten Summenfeldern angelegt. Sollte eine Datei bereits unter dem zu Anfang bestimmten Namen existieren, kann diese gelöscht und neu eingerichtet werden.

Buchen:

Nach Eingabe der Nummer des zu buchenden Kontos wird bestimmt, ob es sich um ein Einnahme oder Ausgabekonto handelt. Das Konto "GEHALT" z.B. ist ein Einnahmekonto und das Konto "MIETE" ein Ausgabekonto.
Danach wird der alte Stand des Kontos ausgegeben. Nun buchen Sie den entsprechenden Betrag der immer positiv ist. Sollte es sich um eine Korrekturbuchung handeln, so geben Sie einen negativen Betrag an.
Nun wird der neue Stand ausgegeben und eine erneute Buchung ermöglicht.

Kontenübersicht

Nach Eingabe der Kontonummer werden die Summen der 12 Monate somwie die Gesamtsumme des Jahres ausgegeben. Somit erhalten Sie einen Überblick über die Ausgaben bzw. Einnahmen eines Kontos in einem Jahr.

Kontennamen ausgeben

Jedes Konto wird mit seiner Nummer bestimmt. Sollte einmal eine Nummer in Vergessenheit geraten, so besteht die Möglichkeit, in diesem Unterprogramm alle Konten mit Nummer und entsprechendem Namen auszugeben.

Monatsübersicht:

Hier werden die Einnahmen bzw. Ausgaben aller Konten in einem Monat ausgegeben. Der Monatssaldo aller Konten schließen dieses Unterprogramm ab

Jahresübersicht:

Dieses Unterprogramm zeigt Ihnen die Jahressummen aller Konten und den Jahressaldo. Dieses Auflisten nimmt etwas Zeit in Anspruch, da alle Monatsfelder jedes Records gelesen und aussummiert werden muß. Es wird also auf die gesamte Datei zugegriffen.

Wir glauben, alle wesentlichen Anforderungen an ein

derartiges Programm erfüllt zu haben. Sollten Sie aber die ein oder andere Idee einer Erweiterung haben, so studieren Sie das Programm mit der anschließenden Dokumentation. Dann werden Ihnen Eingriffe in das Programm zur individuellen Anpassung keine Probleme bereiten.

Das Listing des Programms:

```
100 POKE53280,2:POKE53281,2:PRINTCHR$(158);:
    BL$="                         ":DIMS(12)
110 GOSUB2050
120 INPUT"AKTUELLES JAHR : ";J$
130 IFJ$<"1983"ORJ$>"1999"THENPRINTCHR$(145);:GOTO120
140 GOSUB2050
150 PRINT"FUNKTIONSAUSWAHL:"
160 PRINT"-----------------":PRINT
170 PRINT"      -1- KONTEN ANLEGEN"
180 PRINT"      -2- BUCHEN"
190 PRINT"      -3- KONTENUEBERSICHT"
200 PRINT"      -4- KONTENNAMEN AUSGEBEN"
210 PRINT"      -5- MONATSUEBERSICHT"
220 PRINT"      -6- JAHRESUEBERSICHT":PRINT
230 PRINT"      -0- PROGRAMMENDE"
240 GET X$:IF X$<"0" OR X$>"6" THEN 240
250 IF X$<>"0"THEN270
260 END
270 ON VAL(X$)GOSUB 290,560,920,1160,1370,1720
280 GOTO140
290 REM ====================
300 REM     KONTEN ANLEGEN
310 REM ====================
320 GOSUB2050
330 PRINT"ACHTUNG! EINE EVTL. DATEI DIESES JAHRES"
340 PRINT"WIRD GELOESCHT!":PRINT
350 PRINT"SICHER (J/N)?"
360 GETX$:IFX$<>"J"AND X$<>"N"THEN 360
370 IF X$="J"THEN390
380 CLOSE1:CLOSE2:RETURN
390 OPEN2,8,15,"S:KONTEN"+J$
400 OPEN1,8,2,"KONTEN"+J$+",L,"+CHR$(141)
410 GOSUB2050
420 INPUT"WIEVIELE KONTEN (1-20): ";KZ
430 PRINT
440 IFKZ<1ORKZ>20THENPRINTCHR$(145);:GOTO420
450 FORI=1TOKZ
460 PRINT"NAME KONTO NR.";I;": ";
470 INPUTKN$
480 IFLEN(KN$)>20THENPRINTCHR$(145);:GOTO460
490 RC$=KN$+LEFT$(BL$,20-LEN(KN$))
500 FORX=1TO12
510 RC$=RC$+STR$(0)+LEFT$(BL$,8)
520 NEXTX
530 PRINT#1,RC$
540 NEXT I
550 CLOSE1:CLOSE2:RETURN
```

```
560 REM ==================
570 REM    B U C H E N
580 REM ==================
590 GOSUB2050
600 INPUT"KONTONUMMER";KN
610 IFKN<1ORKN>20THENPRINTCHR$(145);:GOTO600
620 GOSUB2140
630 PRINT"----------------------------"
640 PRINT"NR.";KN;" - ";KN$
650 PRINT"----------------------------"
660 PRINT"EINNAHME ODER AUSGABE (E/A)?"
670 PRINT"----------------------------"
680 GETX$:IFX$<>"E"ANDX$<>"A"THEN680
690 INPUT"MONAT (1-12)   : ";M
700 IFM<1ORM>12THENPRINTCHR$(145);:GOTO690
710 PRINT"----------------------------"
720 PRINT"ALTER STAND    : ";S(M)
730 PRINT"----------------------------"
740 INPUT"BUCHUNGSBETRAG : ";BB
750 PRINT"----------------------------"
760 IFX$="E"THEN S(M)=S(M)+BB:GOTO780
770 S(M)=S(M)-BB
780 PRINT"NEUER STAND    : ";S(M)
790 PRINT"----------------------------"
800 RC$=KN$+LEFT$(BL$,20-LEN(KN$))
810 FORI=1TO12
820 S$=STR$(S(I))
830 RC$=RC$+S$+LEFT$(BL$,10-LEN(S$))
840 NEXTI
850 PRINT#2,"P"+CHR$(2)+CHR$(KN)+CHR$(0)+CHR$(1)
860 PRINT#1,RC$
870 CLOSE1:CLOSE2
880 PRINT"WEITERE BUCHUNGEN (J/N)?"
890 GETX$:IFX$<>"J"ANDX$<>"N"THEN890
900 IFX$="J"THENGOSUB2050:GOTO600
910 RETURN
920 REM ====================
930 REM KONTENUEBERSICHT
940 REM ====================
950 GOSUB2050
960 INPUT"KONTONUMMER : ";KN
970 IFKN<1ORKN>20THENPRINTCHR$(145);:GOTO960
980 GOSUB2140
990 GOSUB2050:PRINTCHR$(145);CHR$(145);
1000 PRINT"-------------------------"
1010 PRINT"NR.";KN;" - ";KN$
1020 PRINT"-------------------------"
1030 PRINT"MONAT  SALDO"
1040 PRINT"-------------------------"
1050 GS=0
1060 FORI=1TO12
1070 PRINTI;TAB(8);S(I)
1080 GS=GS+S(I)
1090 NEXTI
1100 PRINT"-------------------------"
1110 PRINT"GESAMT";TAB(8);GS
```

```
1120 PRINTTAB(8);"======="
1130 PRINT"WEITER MIT RETURN"
1140 INPUTX$
1150 CLOSE1:CLOSE2:RETURN
1160 REM ====================
1170 REM KONTENNAMEN AUSGEBEN
1180 REM ====================
1190 GOSUB2050
1200 OPEN1,8,2,"KONTEN"+J$+",L,"+CHR$(141)
1210 OPEN2,8,15
1220 I=1
1230 PRINT#2,"P"+CHR$(2)+CHR$(I)+CHR$(0)+CHR$(1)
1240 RC$=""
1250 FORX=1TO20
1260 GET#1,X$
1270 RC$=RC$+X$
1280 NEXTX
1290 INPUT#2,X
1300 IFX=50THEN 1340
1320 PRINTI;"  -  ";RC$
1330 I=I+1:GOTO1230
1340 PRINT"WEITER MIT RETURN"
1350 INPUTX$
1360 CLOSE1:CLOSE2:RETURN
1370 REM -----------------
1380 REM  MONATSUEBERSICHT
1390 REM -----------------
1400 GOSUB2050
1410 INPUT"MONAT : ";M
1420 GOSUB2050
1430 PRINT"---------------------------------"
1440 PRINT"NR.  NAME                  BETRAG"
1450 PRINT"---------------------------------"
1460 OPEN1,8,2,"KONTEN"+J$+",L,"+CHR$(141)
1470 OPEN2,8,15
1480 GS=0
1490 FOR KN=1TO20
1500 KN$="":S$=""
1510 PRINT#2,"P"+CHR$(2)+CHR$(KN)+CHR$(0)+CHR$(1)
1520 FOR I=1TO20
1530 GET#1,X$
1540 KN$=KN$+X$
1550 NEXTI
1560 INPUT#2,F
1570 IFF<>50THEN 1590
1580 GOTO1670
1590 PRINT#2,"P"+CHR$(2)+CHR$(KN)+CHR$(0)+CHR$(20+(M-1)*10)
1600 FOR I=1TO10
1610 GET#1,X$
1620 S$=S$+X$
1630 NEXT I
1640 GS=GS+VAL(S$)
1650 PRINT KN;TAB(6);KN$;TAB(26);S$
1660 NEXT KN
1670 PRINT"---------------------------------"
1680 PRINT"GESAMTSALDO";TAB(26);STR$(GS)
```

```
1690 PRINTTAB(26);"======="
1700 PRINT"WEITER MIT RETURN";
1710 INPUTX$:CLOSE1:CLOSE2:RETURN
1720 REM ====================
1730 REM  JAHRESUEBERSICHT
1740 REM ====================
1750 GOSUB2050
1760 OPEN1,8,2,"KONTEN"+J$+",L,"+CHR$(141)
1770 OPEN2,8,15
1780 PRINT"-----------------------------------"
1790 PRINT"NR.  NAME                JAHRESALDO"
1800 PRINT"-----------------------------------"
1810 GS=0
1820 FOR KN=1TO20
1830 PRINT#2,"P"+CHR$(2)+CHR$(KN)+CHR$(0)+CHR$(1)
1840 RC$=""
1850 FORI=1TO140
1860 GET#1,X$
1870 RC$=RC$+X$
1880 NEXTI
1890 INPUT#2,F:IFF=50THEN1980
1900 KN$=LEFT$(RC$,20)
1910 JS=0
1920 FORI=1TO12
1930 JS=JS+VAL(MID$(RC$,20+(I-1)*10,10))
1940 NEXTI
1950 GS=GS+JS
1960 PRINTKN;TAB(6);KN$;TAB(26);JS
1970 NEXTKN
1980 PRINT"-----------------------------------"
1990 CLOSE1:CLOSE2
2000 PRINT"GESAMTSALDO";TAB(26);GS
2010 PRINTTAB(26);"======="
2020 PRINT"WEITER MIT RETURN"
2030 INPUTX$
2040 RETURN
2050 REM ====================
2060 REM    PROGRAMMMKOPF
2070 REM ====================
2080 PRINTCHR$(147);
2090 PRINTTAB(4);"================================"
2100 PRINTTAB(4);"H A U S H A L T S B U C H  "+J$
2110 PRINTTAB(4);"================================"
2120 PRINT:PRINT
2130 RETURN
2140 REM ==================
2150 REM   KONTO EINLESEN
2160 REM ==================
2170 OPEN1,8,2,"KONTEN"+J$+",L,"+CHR$(141)
2180 OPEN2,8,15
2190 PRINT#2,"P"+CHR$(2)+CHR$(KN)+CHR$(0)+CHR$(1)
2200 RC$=""
2210 FORI=1TO140
2220 GET#1,X$
2230 RC$=RC$+X$
2240 NEXT I
```

```
2250 INPUT#2,F
2260 IFF<>50THEN 2300
2270 PRINT"JAHRESDATEI ODER KONTO NICHT GEFUNDEN!":PRINT
2280 PRINT"WEITER MIT RETURN"
2290 CLOSE1:CLOSE2:RETURN
2300 KN$=LEFT$(RC$,20)
2310 GS=0
2320 FORI=1TO12
2330 S(I)=VAL(MID$(RC$,20+(I-1)*10,10))
2340 GS=GS+S(I)
2350 NEXT I
2360 RETURN
```

Die Dokumentation des Programms:

Vorspann:

100	Bildschirm- und Zeichenfarbe setzen; Leerzeichenstring definieren; Variable für Kontensummen dimensionieren.
110-130	Programmkopf anzeigen und aktuelles Jahr einlesen.
140-280	Programmfunktionen anzeigen und Auswahl einlesen; entsprechendes Unterprogramm aufrufen.

Konten anlegen:

390-400	Evtl. vorhandene Datei dieses Jahres löschen und neue Datei eröffnen.
480	Eingebenen Kontennamen in der Position 1-20 des Records RC$ bereitstellen.
500-540	Monatssummen auf Null setzen und als Stringvariablen im Record bereitstellen.
530	Record mit abschließendem RETURN übertragen. RETURN wird standardmäßig von PRINT gesendet.

Buchen:

590	Routine "Konto einlesen" aufrufen. Diese Routine stellt die Monatssummen des Kontos in den Variablen S(1) bis S(12) zur Verfügung.
800	Kontenname in Record übertragen.
810-840	Kontosummen in Record übertragen.
850-860	Record übertragen.

Kontenübersicht:

980	Gewünschtes Konto einlesen und Monatssummen in den Variablen S(1) bis S(12) bereitstellen.
1050-1090	Monatssummen anzeigen und in Gesamtsumme (GS) aufaddieren.
1110	Gesamtsumme anzeigen.

Kontennamen ausgeben:
---------------------:

1220 Kontonummer auf Anfangswert setzen.
1230 Auf Record des entsprechenden Kontos positionieren.
1240-1280 Kontoname aus Record in RC$ einlesen.
1290-1300 Wenn RECORD NOT PRESENT im Fehlerkanal (Fehler 50), dann Routine abbrechen.
1320 Kontonummer und Name ausgeben.

Monatsübersicht:

1490-1660 Schleife zum Einlesen aller Konten.
1510 Auf Record positionieren.
1520-1550 Kontenname einlesen.
1560-1580 Feststellen, ob Konto vorhanden; Abbruch wenn nicht alle 20 Konten definiert wurden.
1590 Positionieren auf Summenfeld des gewünschten Monats.
1600-1630 Einlesen der Monatssumme.
1640 Monatssumme in Gesamtsumme aufaddieren.
1650 Kontonummer, Kontoname und Monatssumme ausgeben.
1680 Gesamtsaldo (Gesamtsumme) ausgeben.

Jahresübersicht:

1820-1970 Schleife zum Einlesen aller Konten.
1830 Auf Record positionieren.
1850-1880 Gesamten Record in RC$ einlesen.
1890 Testen, ob RECORD NOT PRESENT.
1900 Kontoname aus Record holen.
1920-1940 Monatssummen lesen, in numerischer Form umwandeln und in Jahressumme (JS) aufaddieren.
1950 Jahressumme (JS) in Gesamtsumme (GS) aufaddieren.
1960 Kontonummer, Kontoname und Jahressumme ausgeben.
2000 Gesamtsaldo (Monatssaldo) ausgeben.

Konto einlesen:

2190 Auf in KN übergebenen Record positionieren.
2210-2240 Record in RC$ einlesen.
2250-2260 Testen, ob RECORD NOT PRESENT.
2300 Kontoname aus Record lesen.
2320-2350 Monatssummen aus Record lesen, in numerischer Form umwandeln und der Tabelle S(1) bis S(12) übergeben.

1.6 Die Fehlermeldungen der Floppy und ihre Ursachen

Machen Sie bei der Bedienung der Floppy einen Fehler oder tritt ein Disketten- oder sonstiger Fehler auf, so signalisiert dies die Floppy durch Blinken der roten Leuchtdiode (LED) am Laufwerk. Die LED blinkt solange, bis Sie die Fehlermeldung der Floppy gelesen haben oder bis Sie einen neuen Befehl zur Floppy geschickt haben. Als erstes wollen wir sehen, wie man die Fehlermeldung der Floppy einlesen kann.

Dazu muß der Fehler- bzw. Kommandokanal unter der Sekundäradresse 15 geöffnet sein:

```
100 OPEN 15,8,15
110 INPUT#15, A,B$,C,D
120 PRINT A,B$,C,D
```

War keine Fehlerbedingung aufgetreten, so führt dies zur Ausgabe von

```
 0         OK        0         0
```

Dabei bedeutet die erste Zahl (A) die Fehlernummer, in unserem Falle 0; kein Fehler. Als nächstes folgt die Fehlermeldung im Klartext (Variable B$). Die Variablen C und D enthalten die Track- und Sektornummer, bei denen der Fehler aufgetreten ist, sofern dies von der Fehlerart her möglich ist (hauptsächlich bei Hardware-Fehlern und blockorientierten Befehlen).

Eine analoge Routine gibt die Fehlermeldung zusammenhängend wieder:

```
100 OPEN 15,8,15
110 GET#15,A$ : PRINT A$; : IF ST<>64 THEN 110

00, OK,00,00
```

Hier werden solange Zeichen vom Fehlerkanal geholt und ausgegeben, bis das Ende erkannt wird (Status = 64). Dies gibt die Fehlermeldung genauso wieder, wie dies mit dem BASIC 4.0 Befehl

```
PRINT DS$
```

möglich ist. Hier sind DS$ und DS reservierte Variablen, die die komplette Fehlermeldung bzw. die Fehlernummer enthalten. Jeder Bezug auf diese Variablen gibt den Fehlerstatus der letzten Diskettenoperation wieder.

Auf den nächsten Seiten sind nun alle möglichen Fehlermeldungen in Detail beschrieben.

00, OK,00,00
Diese Meldung tritt dann auf, wenn die letzte Diskettenoperation fehlerfrei verlaufen ist oder falls nach dem Lesen der letzten Fehlermeldung keine Daten oder kein Befehl zur Floppy geschickt wurden.

01,FILES SCRATCHED,XX,00
Dies ist die Rückmeldung nach einem SCRATCH-Befehl. Die Zahl XX gibt dabei an, wieviel Dateien gelöscht wurden, da z.B. durch die Verwendung des Jokers mit einem Befehl mehr als eine Datei gelöscht werden kann. Da dies keine eigentliche Fehlermeldung ist, blinkt dabei auch nicht die LED. Arbeiten Sie mit dem BASIC 4.0 Befehl 'SCRATCH', so wird die Rückmeldung automatisch geholt und angezeigt.

20,READ ERROR,TT,SS
Dieser Fehler bedeutet, daß der 'Header' (Kopf) eines Blocks nicht gefunden wurde. Dabei handelt es sich meist um eine defekte Diskette. TT und SS bezeichnen hier Track und Sektor, bei dem der Fehler aufgetreten ist. Maßnahmen: Defekte Diskette auswechseln.

21,READ ERROR,TT,SS
Auch dies ist ein Lesefehler. Hier wurde zu einem Block die entsprechende SYNC (Synchron-) Markierung nicht gefunden. Als Ursache hier kann keine oder eine nicht formatierte Diskette sein. Dieser Fehler kann auch auf einen dejustierten Schreib/Lesekopf hindeuten. Maßnahmen: Entweder Diskette austauschen, formatieren oder Schreib/Lesekopf justieren lassen.

22,READ ERROR,TT,SS
Diese Fehlermeldung bedeutet einen Prüfsummenfehler im Header eines Datenblocks, der durch fehlerhaftes Schreiben eines Blocks verursacht sein kann.

23,READ ERROR,TT,SS
Bei diesem Lesefehler konnte ein Datenblock zwar in den DOS-Puffer gelesen werden, es wurde jedoch ein Prüfsummenfehler festgestellt. Ein oder mehrere Datenbytes sind fehlerhaft. Maßnahmen: Files so weit wie möglich auf eine andere Diskette "retten".

24,READ ERROR,TT,SS
Auch bei dieser Fehlermeldung handelt es sich um einen Prüfsummenfehler entweder im Datenblock oder im vorausgehenden Datenheader. Es wurden fehlerhafte Bytes eingelesen. Maßnahmen: wie Fehler 23.

25,WRITE ERROR,TT,SS
Dieser Fehler ist eigentlich ein VERIFY ERROR. Nach jedem Schreiben eines Datenblocks werden die Daten noch einmal gelesen und mit den Daten im Puffer verglichen. Bei fehlender Übereinstimmung wird dieser Fehler gemeldet. Maßnahmen: Befehl, der den Fehler verursachte wiederholen. Falls kein Erfolg, dann entsprechenden Datenblock mit

BLock-Allocate für weitere Beutzung sperren.

26,WRITE PROTECT ON,TT,SS
Es wurde der Versuch unternommen, auf eine Diskette zu schreiben, die einen Schreibschutzaufkleber enthält. Maßnahmen: Schreibschutz entfernen.

27,READ ERROR,TT,SS
Hier handelt es sich um einen Prüfsummenfehler im Header eines Datenblocks. Maßnahmen: Befehl wiederholen oder Block sperren.

28,WRITE ERROR,TT,SS
Nach dem Schreiben eines Datenblocks wird die SYNC (Synchron-) Zeichenfolge des nächsten Datenblocks nicht gefunden. Maßnahmen: Diskette neu formatieren oder austauschen.

29,DISK ID MISMATCH,TT,SS
Die ID (zweistellige Diskettenidentifikation) im DOS-Speicher stimmt nicht mit der ID auf der Diskette überein. Die Diskette wurde entweder nicht initialisiert oder es liegt ein Fehler im Header eines Datenblocks vor. Maßnahmen: Diskette initialisieren.

30,SYNTAX ERROR,00,00
Ein Befehl, der über den Kommandokanal geschickt wurde, kann vom DOS nicht interpertiert werden. Maßnahmen: Befehl überprüfen und korrigieren.

31,SYNTAX ERROR,00,00
Ein Befehl wird vom DOS nicht erkannt, z.B. BACKUP-Befehl (Duplicate) auf der 1541. Maßnahmen: Ausweichbefehl/programm verwenden.

32,SYNTAX ERROR,00,00
Der über den Kommandokanal gesandte Befehl ist länger als 40 Zeichen. Maßnahmen: Befehl verkürzen.

33,SYNTAX ERROR,00,00
Beim OPEN- oder SAVE-Befehl wurde der Joker ('*', '?') unzulässig verwendet. Maßnahmen: Joker entfernen.

34,SYNTAX ERROR,00,00
Das DOS kann den Filenamen in einem Befehl nicht finden, weil z.B. der Doppelpunkt ':' nach dem Befehlswort vergessen wurde. Maßnahmen: Befehl überprüfen.

39,FILE NOT FOUND,00,00
Benutzerprogramm vom Typ 'USR' zum automatischen Ausführen wurde nicht gefunden. Maßnahmen: Filenamen überprüfen.

50,RECORD NOT PRESENT,00,00
Bei einer relativen Datei wurde ein Datensatz angesprochen, der noch nicht geschrieben wurde. Beim Schreiben eines Datensatzes ist dies kein eigentlicher Fehler,

sondern weist nur darauf hin, daß ein neuer Datensatz angelegt wird. Sie können diese Fehlermeldungen vermeiden, wenn Sie beim Anlegen einer relativen Datei direkt in den Datensatz mit der höchsten Nummer CHR$(255) schreiben. Bei weiteren Zugriffen kommt dieser Fehler dann nicht mehr vor.

51,OVERFLOW IN RECORD,00,00
Beim Schreiben eines Datensatzes in eine relative Datei ist die Anzahl der Zeichen (einschließlich des Carriage Return) größer als die Datensatzlänge der Datei. Die überzähligen Zeichen werden ignoriert.

52,FILE TOO LARGE,00,00
Die Datensatznummer einer relativen Datei ist zu groß; für das Anlegen dieses Datensatzes reicht die freie Diskettenkapazität nicht mehr aus. Maßnahmen: Andere Diskette verwenden oder Recordanzahl verringern.

60,WRITE FILE OPEN,00,00
Es wurde versucht, eine Datei zum Lesen zu öffnen, die beim Schreiben nicht geschlossen wurde, weil z.B. die Diskette aus dem Laufwerk genommen wurde, ehe die geöffnete Datei geschlossen wurde. Maßnahmen: Modus 'M' im OPEN-Befehl zum Auslesen dieser Datei verwenden.

61,FILE NOT OPEN,00,00
Es wurde eine Datei angesprochen, die nicht geöffnet war. Maßnahmen: Datei öffnen oder Dateiname überprüfen.

62,FILE NOT FOUND,00,00
Es wurde versucht, ein Programm zu laden oder eine Datei zu öffnen, die nicht auf der Diskette existiert. Maßnahmen: Filename überprüfen.

63,FILE EXISTS,00,00
Der Versuch, eine neue Datei mit einem Namen anzulegen, der schon auf der Diskette existiert, führt zu dieser Fehlermeldung. Maßnahmen: Anderen Filenamen oder Klammeraffe verwenden.

64,FILE TYPE MISMATCH,00,00
Der Dateityp beim Öffnen einer Datei stimmt nicht mit dem Dateityp im Directory überein. Maßnahmen: Filetyp korrigieren.

65,NO BLOCK,TT,SS
Diese Fehlermeldung wird beim BLOCK-ALLOCATE Befehl ausgegeben, wenn der zu belegende Block nicht mehr frei war. Das DOS sucht in diesem Falle selbsttätig einen freien Block mit höherer Sektor- und/oder Tracknummer und gibt diese Werte als Track- und Sektornummer der Fehlermeldung aus. Ist kein Block mit größerer Nummer mehr frei, wird zweimal 0 ausgegeben.

66,ILLEGAL TRACK OR SECTOR,TT,SS

Wenn man bei den Blockbefehlen sich auf nicht existierende Blocks bezieht, wird diese Fehlermeldung ausgegeben.

67,ILLEGAL TRACK OR SECTOR,TT,SS
Die Track-Sektor-Verkettung einer Datei zeigt auf einen nicht existierenden Track oder Sektor.

70,NO CHANNEL,00,00
Es wurde versucht, mehr Dateien zu öffnen als Kanäle vorhanden sind oder ein Direktzugriffskanal ist schon belegt.

71,DIR ERROR,TT,SS
Die Anzahl der freien Blocks im DOS-Speicher stimmt mit dem Bitmuster der BAM nicht überein. Evtl. wurde die Diskette nicht initialisiert.

72,DISK FULL,00,00
Auf der Diskette sind nur noch weniger als 3 Blocks frei oder die maximale Anzahl an Directoryeinträgen wurde erreicht (144 auf der VC 1541).

73,CBM DOS V2.6 1541,00,00
Diese Meldung erscheint als Einschaltmeldung der VC 1541. Als Fehlermeldung tritt sie auf, wenn versucht wird, auf eine Diskette zu schreiben, die nicht mit der gleichen DOS-Version formatiert wurde, z.B. mit dem Vorläufer der CBM 4040, der CBM 3040 (DOS Version 1.0).

74,DRIVE NOT READY,00,00
Wenn man versucht, die Floppy anzusprechen, ohne daß eine Diskette im Laufwerk liegt, erhält man diese Fehlermeldung.

75,FORMAT SPEED ERROR,00,00
Diese Fehlermeldung gibt es nur auf der CBM 8250. Sie zeigt Abweichungen von der Normdrehzahl während der Formatierung an.

1.7 Übersicht aller Befehle mit Vergleich BASIC 2.0 - BASIC 4.0 - DOS 5.1

BASIC 2.0	BASIC 4.0 (Abk.)	DOS 5.1
OPEN - Modus 'A'	APPEND (aP)	
	BACKUP (bA)	
LOAD"$",8 & LIST	CATALOG (cA)	@$ oder >$
V(alidate)	COLLECT (coL)	@V oder >V
	CONCAT (conC)	
C(opy)	COPY (coP)	@C:.. oder >C:..
CLOSE ...	DCLOSE (dC)	
LOAD"...",8	DLOAD (dL)	↑file oder /file
		(↑ = DLOAD & RUN)
OPEN ...,8,...	DOPEN (dO)	
OPEN 1,8,15	DS$, DS	@ oder >
SAVE"...",8	DSAVE (dS)	
N(ew)	HEADER (hE)	@N:.. oder >N:..
I(nitialise)	I(nitialise)	@I oder >I
P	RECORD (reC)	
R(ename)	RENAME (reN)	@R:.. oder >R..
S(cratch)	SCRATCH (sC)	@S:.. oder >S..

Diese Tabelle stellt die verschiedenen BASIC-Versionen gegenüber. Das DOS 5.1 befindet sich auf der TEST/DEMO-Diskette und wird im Kaptitel 4.2.1 beschrieben. Der wesentliche Unterschied zwischen BASIC 2.0 und BASIC 4.0 ist, daß mit BASIC 2.0 jeder Befehl, der vom Disketten-Betriebssystem (DOS) ausgeführt wird, über den Kanal 15 gesendet werden muß. Die Disketten-Befehle des BASIC 4.0 jedoch verwalten diesen Kanal selbstständig (mit Ausnahme von INITIALISE). So erzeugt dieses BASIC z.B. aus dem Befehl HEADER D0,"DISK1",IHJ die gleiche Befehlsfolge, die vom BASIC 2.0 dazu angegeben werden muß, nämlich:

```
OPEN 1,8,15,"N:DISK1,HJ"
CLOSE 1
```

Doch nun die Erklärung der BASIC 4.0-Befehle:

Beachten Sie die folgenden Parameter:

- lfn = logische Filenummer
- dn = Drivenummer - bei Doppellaufwerken gibt es ein Drive 0 (D0) und ein Drive 1 (D1); Singlelaufwerke werden mit D0 adressiert.
- ga = Geräteadresse der Diskettenstation (U4 bis U31)

Angaben in Klammern brauchen nicht angegeben werden. Dann werden die Standardparameter D0 und U8 eingesetzt.

APPEND:

Dieser Befehl ermöglicht das Anhängen von Datensätzen an eine sequentielle Datei, wie es in BASIC 2.0 mit dem OPEN-Modus 'A' realisiert wird.
Dieser Befehl hat das folgende Format:

```
APPEND#lfn,"dateiname"(,Ddn,Uga)
```

Soll z.B die sequentielle Datei "SEQU.l", die sich auf Drive 0 befindet, um einen Datensatz erweitert werden, so ist dazu die folgende Befehlsfolge notwendig:

```
100 APPEND#1,"SEQU.1",D0
110 PRINT#1,X$
120 CLOSE 1
```

BACKUP:

Mit diesem Befehl kann eine gesamte Diskette kopiert werden. Der BACKUP-Befehl ist jedoch nur bei Doppellaufwerken einsetzbar. Beachten Sie das Format dieses Befehls:

```
BACKUP Ddn TO Ddn(,Uga)
```

Wichtig ist, daß entweder D0 TO D1 oder D1 TO D0 angegeben werden muß. Ein Beispiel:

Es soll eine Kopie der Diskette in Drive 1 auf die Diskette in Drive 0 erstellt werden. Dazu wird folgender Befehl eingegeben:

```
BACKUP D1 TO D0
```

CATALOG:

Der CATALOG-Befehl des BASIC 4.0 hat den Vorteil, daß das Anzeigen des Disketteninhaltes nicht den BASIC-Speicher löscht, wie es beim BASIC 2.0 der Fall ist. Das Format des Befehls:

```
CATALOG (Ddn,Uga)
```

Wird bei Doppellaufwerken keine Drivenummer angegeben, so werden die Inhalte beider Disketten ausgegeben. Bei Singlelaufwerken wird CATALOG D0 erzeugt. Ein Beispiel:

```
CATALOG D0
```

Es wird das Inhaltsverzeichnis der Diskette in Drive 0 ausgegeben.

COLLECT:

Dieser Befehl entspricht dem VALIDATE-Befehl des BASIC 2.0. Die Syntax des Befehls sieht so aus:

```
COLLECT (Ddn)
```

CONCAT:

CONCAT verkettet sequentielle Files, indem einem File die Daten eines zweiten Files angehängt werden. Das Format:

```
CONCAT (Ddn,)"filel" to (Ddn,)"file2" (ON Uga)
```

Angenommen Sie wollen die Daten der Datei "SEQU.2" in Drive 0 an die Datei "SEQU.1" in D1 anhängen. Um dies zu erreichen geben Sie folgenden Befehl ein:

```
CONCAT D0,"SEQU.2" TO D1,"SEQU.1"
```

COPY:

Mit diesem Befehl können Files (ausgenommen relative Files) von einem Drive auf das andere kopiert werden. Somit findet der Befehl bei Singlelaufwerken keine Anwendung. Die Syntax des Befehls sieht folgendermaßen aus:

```
COPY (Ddn,)("filel") TO (Ddn,)("file2")
```

Sollen alle Files übernommen werden (z.B. von Drive 0 auf Drive 1), so reicht die die folgende Befehlsform aus:

```
COPY D0 TO D1
```

DCLOSE:

Der Befehl DCLOSE hat dieselbe Funktion wie der einfache CLOSE-Befehl, mit folgenden Ausnahmen:

```
DCLOSE          schließt alle Files
DCLOSE#1        schließt das File mit der Nummer 1
DCLOSE#1 ON U9  schließt das logische File #1 der Geräte-
                adresse 9
DCLOSE U8       schließt alle Files der Geräteadresse 8
```

Der Befehl hat die folgende Syntax:

```
DCLOSE (#lfn) (ON Uga)
```

DLOAD:

Der Befehl DLOAD hat den Vorteil, daß standardmäßig von Geräteadresse 8 geladen wird. Das Format:

```
DLOAD "programm" (,Ddn)(,Uga)
```

Wollen Sie z.B. das Programm "PRG.2" von Drive 0 laden oder von einem Einzellaufwerk laden, so geben Sie den Befehl

```
DLOAD "PRG.2"
```

ein. Drive 0 (D0) wird standardmäßig eingesetzt.

DOPEN:

Dieser Befehl des BASIC 4.0 ist sehr umfangreich. Das folgende Format bestätigt es:

```
DOPEN#lfn,"file"(,Ddn)(,Uga)(,fileparamter)
```

Das Besondere an dieser Art des Öffnens ist der Fileparameter. Es gibt zwei Fileparameter, die folgende Funktion haben:

'L'-Parameter:	'W'-Parameter	Wirkungsweise
JA	NEIN	Eine relative Datei wird geöffnet.
NEIN	JA	Ein sequentielles File wird zum Schreiben geöffnet.
NEIN	NEIN	Ein File wird zum lesen geöffnet. (REL,SEQ,PRG,USR)

Zusätzlich zum 'L'-Parameter muß die Recordlänge angegeben werden (z.B. L80). Ein derartiger DOPEN-Befehl sieht dann so aus:

```
DOPEN#1,"FILE.REL",D0,L80
```

Hier wird ein relatives File mit einer Recordlänge von 80 Bytes erstmals geöffnet. Die Angabe der Fileparameter ist nur einmal, und zwar bei der Erstellung des Files erforderlich. Alle späteren Öffnungen des Files können ohne Parameterangabe erfolgen.

DS$ & DS:

Nach Auftreten eines Diskettenfehlers kann entweder die gesamte Fehlermeldung mit PRINT DS$ oder nur die Fehlernummer mit PRINT DS angezeigt werden. Selbstverständlich kann auch innerhalb eines Programms der Fehler abgefragt und dementsprechend verzweigt werden. Z.B.:

```
100 IF DS = 26 THEN GOTO.....
```

DSAVE:

Mit diesem Befehl können Programme auf Diskette gespeichert werden. Das folgende Format ist zu beachten:

```
DSAVE (Ddn,)"programmname"(,Uga)
```

HEADER:

Mit dem HEADER-Befehl werden im BASIC 4.0 Disketten formatiert. Er entspricht dem NEW-Befehl im BASIC 2.0. Die Syntax des Befehls:

```
HEADER "diskettenname",D0,Iid(,Uga)
oder   HEADER Ddn,"diskettenname",Iid
```

Hier gibt es zwei Möglichkeiten, das Laufwerk zu bestimmen. Die Angabe id ist die Disketten-Identifikation. Wird sie nicht angegeben, so wird der Disketten, vorausgesetzt sie ist formatiert, lediglich ein neuer Name zugewiesen und alle darauf befindlichen Files gelöscht.

RECORD:

Dieser Befehl entspricht dem Positionier-Befehl des BASIC 2.0, bzw. des DOS 2.6. Mit dem RECORD-Befehl kann also auf einen Record in einer relativen Datei positioniert werden, ohne daß diese Positionierung über Kanal 15 gesendet werden muß. Die Syntax dieses Befehls verdeutlicht, wie komfortabel diese Positionierung ist:

```
RECORD#lfn,rn(,bp)
```

Die logische Filenummer bezieht sich auf das geöffnete, relative File. Für 'rn' wird die Recordnummer (1-65535) und für 'bp' evtl. die Position innerhalb dieses Records (1-254) angegeben.

Ein Beispiel: Sie wollen auf das 12. Byte des 128. Records

einer mit der logischen Filenummer 2 geöffneten, relativen Files positionieren. Der folgende Befehl ermöglicht dies:

```
RECORD#2,128,12
```

RENAME:

Dieses RENAME ist ähnlich dem RENAME des BASIC 2.0. Das Format dieses Befehls:

```
RENAME (Ddn,)"alter name" TO "neuer name"(,Uga)
```

SCRATCH:

Diese Methode des Löschens von Files ist wesentlich komfortabler, denn es kann mit einem Befehl gelöscht werden. Das Format dieses Befehls:

```
SCRATCH (Ddn,)"file"(,Uga)
```

Nach Eingabe eines SCRATCH-Befehls wird mit der Meldung "ARE YOU SURE?" noch einmal eine Annulierung des Befehls ermöglicht. Soll das File wirklich gelöscht werden, so geben Sie 'Y', ansonsten 'N' ein. Nach dem Löschen des Files erscheint die Meldung "FILES SCRATCHED" auf dem Bildschirm.

Kapitel 2: Programmierung für Fortgeschrittene

2.1 Der Direktzugriff auf jeden Block der Diskette

Bei der Handhabung von Dateien und Programmen auf der Floppy, wie sie in Kapitel 1 beschrieben ist, brauchen wir uns um die Organisation auf der Diskette nicht zu kümmern, das Floppybetriebssystem (DOS) erledigt dies automatisch für uns.

Das DOS bietet jedoch auch die Möglichkeit, jeden Block auf der Diskette, der durch Track (Spur) und Sektor bestimmt ist, einzeln anzusprechen. Damit stehen uns jetzt weitreichende Möglichkeiten zur Verfügung, von der Manipulation einzelner Files bis zur Realisierung eigener neuer Dateistrukturen.

Um auf einen Block direkt zugreifen zu können, muß vom DOS ein Datenkanal und ein Datenpuffer zugeordnet werden, über den die Daten übermittelt werden. Der Datenpuffer dient zur Zwischenspeicherung der Daten, ehe sie auf Diskette geschrieben werden, bzw. in den sie von der Diskette gelesen werden. Um dem DOS mitzuteilen, daß wir im Direktzugriff arbeiten wollen, wird ein spezieller Filename im OPEN-Befehl benutzt:

```
OPEN 1,8,2, "#"
```

Mit diesem Befehl wird der logischen Filenummer 1 auf dem Gerät 8, der Floppy, eine Direktzugriffsdatei zugeordnet. Zur Datenübermittlung dient der Kanal 2 der Floppy. An Kanalnummern (Sekundäradresse beim OPEN-Befehl) stehen Ihnen 2 bis 14 zur Verfügung. 0 und 1 sind für LOAD und SAVE reserviert, 15 ist der Kommandokanal. Welche Sekundäradresse Sie wählen, hat keine weitere Bedeutung. Natürlich dürfen Sie eine Sekundäradresse nicht mehrmals verwenden, da das DOS beim zweiten OPEN-Befehl mit gleicher Sekundäradresse die vorherige Datei mit dieser Kanalnummer schließt. Das gilt natürlich auch beim Arbeiten mit normalen Dateien.

Bei dieser Form des OPEN-Befehls sucht die Floppy selbst einen freien Datenpuffer und weist ihn dem angesprochenen Kanal zu. Wir können die Puffernummer lesen, wenn wir unmittelbar nach dem OPEN-Befehl mit GET ein Zeichen abholen. Dieser Wert enthält die Puffernummer.

```
100 OPEN 1,8,2, "#"
110 GET#1, A$
120 PRINT ASC(A$+CHR$(0))
RUN

 3
```

In unserem Falle wurde also Puffer 3 belegt. Die Numerierung der Puffer geht von 0 bis 4. Die Puffer belegen jeweils 256 Byte (wie jeder Block auf der Diskette) und liegen bei der VC

1541 in folgenden Speicherbereichen:

Puffernummer	Speicherbereich
0	$300 - $3FF, 768 - 1023
1	$400 - $4FF, 1024 - 1279
2	$500 - $5FF, 1280 - 1535
3	$600 - $6FF, 1536 - 1791
4	$700 - $7FF, 1792 - 2047

Puffer 4 steht normalerweise nicht zur Verfügung, da dort die BAM gespeichert ist. Arbeiten wir gleichzeitig noch mit normalen Dateien, kann auch Puffer 3 nicht benutzt werden, da er dann fürs Directory benutzt wird. Wollen wir beim Direktzugriff einen bestimmten Pufferspeicher zuordnen, so können wir dies beim OPEN-Befehl mit angeben.

```
OPEN 1,8,2, "#3"
```

Hiermit wird dem Kanal 2 der Puffer 3 ($600 - $6FF) zugeordnet, sofern er noch frei ist. Falls nicht aus besonderen Gründen ein bestimmter Puffer erforderlich ist (z.B. wenn ein ausführbares Maschinenprogramm dort stehen soll), so sollte man dem DOS die Wahl des Puffers überlassen, da bei der Auswahl eines festen Puffers die Möglichkeit, daß er belegt ist, größer ist.

Sie sollten daher nach dem Öffnen des Kanal in jedem Falle den Fehlerkanal abfragen.

```
130 OPEN 15,8,15
140 GET#15, A$ : PRINT A$; : IF ST <> 64 THEN 140
```

Ist der Puffer bereits belegt, so bekommen Sie die Fehlermeldung

```
70,NO CHANNEL,00,00
```

Haben Sie keine anderen Dateien offen, so können Sie bis zu 4 Kanäle für den Direktzugriff öffnen. Es werden dann in der Reihenfolge des Öffnens die Puffer 3 bis 0 zugeordnet, wie Sie folgendem Beispiel entnehmen können.

```
 10 OPEN 1,8,15,"I0" : I=2 : REM FEHLERKANAL
 20 OPEN 2,8,2, "#" : GOSUB 100
 30 OPEN 3,8,3, "#" : GOSUB 100
 40 OPEN 4,8,4, "#" : GOSUB 100
 50 OPEN 5,8,5, "#" : GOSUB 100
 60 OPEN 6,8,6, "#" : GOSUB 100
 70 END
100 GET#I, A$: PRINT ASC(A$+CHR$(0))
110 I=I+1 : REM PUFFERNUMMER
120 GET#1, A$ : PRINT A$; : IF ST <> 64 THEN 120
130 RETURN

 3
00, OK,00,00
```

```
 2
00, OK,00,00
 1
00, OK,00,00
 0
00, OK,00,00
 199
70,NO CHANNEL,00
```

Wie Sie sehen, scheiterte der Versuch, einen 5. Kanal für den Direktzugriff zu öffnen.

Die Datenübertragung von und zu den Pufferspeichern geschieht wie üblich mit GET# bzw. INPUT# und PRINT#-Befehlen.

Hier noch eine Bemerkung zum Einlesen von Daten in den Rechner.

Handelt es sich im Puffer um reine alphanumerische Daten, z.B. Texte, die nicht länger als 88 Zeichen sind und die mit CR (Carriage Return, CHR$(13)) von einander getrennt sind, so können sie ohne weiteres mit INPUT# gelesen werden. Sind jedoch auch Steuerzeichen enthalten oder sind die Texte mit Komma oder Doppelpunkt getrennt, so versagt der INPUT#-Befehl. Hier müssen wir auf den GET#-Befehl ausweichen, der immer nur ein Zeichen holt. Hier müssen wir jedoch beachten, daß mit GET# kein Nullbyte CHR$(0) gelesen werden kann. In diesem Falle erhalten Sie den Leerstring zurück, so daß dies extra abgefragt werden muß, z.B.

```
100 GET#2, A$ : IF A$="" THEN A$ = CHR$(0)
```

Eine andere und meist einfachere Alternative ist die Benutzung des Befehls 'INPUT*', wie er in Kapitel 4.3.1 beschrieben ist. Hier können Sie angeben, wieviel Zeichen in einen String eingelesen werden sollen. Auch gibt es hier keine Probleme mit Nullbytes (CHR$(0)). Hier können wir auch fast den ganzen Puffer (255 Zeichen sind möglich, d.h. bis auf ein Zeichen) mit einem Befehl lesen.

In den nächsten Abschnitten sind nun alle Befehle im Zusammenhang mit dem Direktzugriff ausführlich beschrieben.

Haben Sie sich mit den Block-Befehlen bereits näher befaßt und wollen Sie sich einzelne Blocks komplett auf dem Bildschirm ansehen oder ändern, so können Sie dafür den Disk-Monitor aus Kapitel 4.6 benutzen, der dies auf einfache und komfortable Weise ermöglicht.

2.2 Die Direktzugriffsbefehle

2.2.1 Der Block-Read-Befehl B-R

Der Block-Read-Befehl dient zum Lesen eines Block von Diskette in den Puffer einer zuvor geöffneten Direktzugriffsdatei. Sämtliche Block-Befehle werden über den Kommandokanal (Sekundäradresse 15) an die Floppy geschickt. Der Befehl zum Lesen eines Blocks lautet 'B-R'. Da mit diesem Befehl jedoch das erste Byte eines Block nicht gelesen wird, benutzt man zum Lesen eines Blocks nur den Befehl 'U1'. Der Befehl hat folgende Syntax:

```
U1 Kanalnummer Drive Track Sektor
```

Dabei müssen Sie die Kanalnummer angeben, die Sie beim Öffnen der Direktzugriffsdatei verwendet haben. Als nächstes folgt die Drivenummer; bei der VC 1541 immer Null und dann die Nummern des Tracks und Sektors, den Sie lesen wollen.

```
10 OPEN 1,8,15
20 OPEN 2,8,2, "#"
30 PRINT#1, "U1 2 0 18 0"
```

Damit haben Sie den Inhalt von Track 18 Sektor 0 in den zu Kanal 2 gehörenden Puffer gelesen. Nun können Sie mit GET#2 Daten aus diesem Puffer lesen.

```
40 GET#2, A$,B$
50 PRINT ASC(A$), ASC(B$)

18          1
```

Damit haben wir die beiden ersten Byte aus dem Puffer gelesen und angezeigt. Der Track 18, Sektor 0 enthält die BAM der 1541; die beiden gelesenen Werte bezeichnen den Track und den Sektor des ersten Directory-Blocks.

Im Demo-Programm 'DISPLAY T&S' auf der Testdiskette (Kapitel 4.2.7) wurde dieser Befehl benutzt, um die BAM von Diskette zu lesen und die Belegung der einzelnen Sektoren auf dem Bildschirm grafisch darzustellen.

Mit dem GET#-Befehl können wir so alle 256 Byte des Blocks aus dem Puffer lesen; in unserem Beispiel lesen wir ab Position 144 den Diskettenamen und die ID.

Da die einzelnen Blocks einer Datei so verkettet sind, daß die ersten beiden Bytes auf einem Block jeweils die Track- und Sektornummer des nachfolgenden Blocks enthalten, kann man so den Verlauf einer Datei über die Diskette verfolgen. Die Datei ist dann zuende, wenn man als Folgetrack den Wert Null erhält; das zweite Byte gibt dann an, wieviel Bytes auf diesem Sektor noch zur Datei gehören. Den ersten Sektor einer

Datei kann man mit unserem Programm aus Kapitel 4.1.1 erfahren. Dann kann folgendes kleine Programm alle weiteren Tracks und Sektoren anzeigen, die durch eine Datei belegt sind.

```
100 OPEN 1,8,15
110 OPEN 2,8,2, "#"
120 INPUT "TRACK UND SECTOR ";T,S
130 PRINT# 1, "U1 2 0";T;S
140 GET# 2, T$, S$
150 T = ASC(T$+CHR$(0)): S = ASC(S$+CHR$(0))
160 IF T = 0 THEN CLOSE 2 : CLOSE 1 :END
170 PRINT "TRACK";T ,"SECTOR";S
180 GOTO 130
```

Geben Sie 18 und 0 als Track und Sektor an, so verfolgen Sie die Blöcke für BAM und Directory.

2.2.2 Der Buffer-Pointer-Befehl B-P

Benötigen wir in unserem obigen Beispiel nur den Diskettennamen, der in Track 18 Sektor 0 ab Position 144 steht, so mußten wir nach obiger Methode die ersten 143 Byte überlesen, ehe wir den Namen erhielten. Um den Zugriff auf jedes beliebige Byte zu erleichtern hat man den Block-Pointer-Befehl eingeführt. Damit läßt sich der Zeiger, der die augenblickliche Lese- oder Schreibposition im Puffer angibt, auf jedes beliebige Byte im Puffer setzen. Die Syntax ist folgende:

```
B-P Kanalnummer Position
```

Jetzt kann man den Diskettennamen direkt lesen:

```
100 OPEN 1,8,15
110 OPEN 2,8,2, "#"
120 PRINT# 1, "U1 2 0 18 0"
130 PRINT# 1, "B-P 2 144"
140 FOR I = 1 TO 16 :REM MAXIMALE LAENGE
150 GET# 2, A$ : IF A$=CHR$(160) THEN 170
160 PRINT A$; : NEXT
170 CLOSE 2 : CLOSE 1
```

Hier haben wir nach dem Einlesen des Blocks den Pufferzeiger auf 144 gesetzt und lesen dann 16 Bytes, falls vorher nicht CHR$(160) ('Shift Space') gefunden wurde, welches das Ende des Namens anzeigt.
Die Bytes im Puffer sind von 0 bis 255 nummeriert, das erste Byte hat also die Nummer 0. Beim Lesen eines Blocks mit U1 wird der Pufferzeiger automatisch auf das Byte Nummer null gesetzt. Beim Bewegen des Pufferzeigers ist man völlig frei. Man kann z.B. in unserem obigen Beispiel nach dem Lesen des Namens das Byte Nummer 2 lesen, kann dies einfach durch Setzen des Pufferzeigers auf dieses Byte geschehen.

```
PRINT# 1, "B-P 2 2"
```

2.2.3 Der Block-Write-Befehl B-W

Der Block-Write-Befehl ermöglicht es uns, den Inhalt des Pufferspeichers in einen beliebigen Block auf Diskette zu schreiben. Man kann damit Daten, die man in den Puffer geschrieben hat, auf einen Block der Diskette schreiben. Ebenso ist es möglich, mit dem Block-Read-Befehl einen Block in den Puffer zu lesen, dann einige Bytes zu verändern und den Block dann wieder zurück zu schreiben. Der Block-Write-Befehl wird mit B-W abgekürzt. Da dieser 'B-W'-Befehl jedoch in das erste Byte des Puffers den augenblicklichen Inhalt des Pufferzeigers schreibt, benutzt man hier meist den 'U2'-Befehl. Die Syntax des Befehls ist analog zum B-R Befehl

```
U2 Kanalnummer Drive Track Sektor
```

```
100 OPEN 1,8,15
110 OPEN 2,8,2, "#"
120 PRINT# 2, "TESTDATEN"
130 PRINT# 1, "U2 2 0 1 0"
140 CLOSE 2 :CLOSE 1
```

Hier wird der Text "TESTDATEN" in den zu Kanal zwei gehörenden Puffer geschrieben und dieser dann auf Track 1 Sektor 0 der Diskette. Durch den 'U1'-Befehl werden der Inhalt des Puffers sowie der Pufferzeiger nicht verändert. Wir wollen jetzt den Block-Write-Befehl dazu benutzen, den Namen der Diskette, den wir im letzten Abschnitt gelesen haben, zu ändern. Dazu müssen wir den neuen Namen bis auf eine Länge von 16 Zeichen mit 'Shift Space' CHR$(160) auffüllen, ehe wir ihn auf Diskette schreiben können. Wir benutzen wieder den Buffer-Pointer-Befehl, um den Zeiger direkt auf die gewünschte Position innerhalb des Puffers zu setzen.

```
100 OPEN 1,8,15
110 OPEN 2,8,2, "#"
120 PRINT# 1, "U1 2 0 18 0"
130 PRINT# 1, "B-P 2 144"
140 A$ = "DIREKTZUGRIFF"
150 IF LEN(A$) < 16 THEN A$ = A$+CHR$(160) : GOTO 150
160 PRINT# 2, A$;
170 PRINT# 1, "U2 2 0 18 0"
180 CLOSE 2
190 PRINT# 1, "I0" : CLOSE 1
```

Wir lesen also erst Track 18 Sektor 0 in den Puffer, setzen den Pufferzeiger auf die Position des Diskettennamens und schreiben den auf 16 Zeichen aufgefüllten Namen in den Puffer. Jetzt wird in Zeile 170 der Pufferinhalt wieder in der ursprünglichen Block geschrieben und der Kanal 2 geschlossen. Dann wird die Diskette neu initialisiert, damit

BAM und Name in den DOS-Speicher übernommen werden. Holen Sie jetzt das Inhaltsverzeichnis mit

```
LOAD "$",8
LIST
```

auf den Bildschirm, so sehen Sie, daß unsere Diskette einen neuen Namen bekommen hat.

2.2.4 Der Block-Allocate-Befehl B-A

Der Block-Allocate-Befehl hat die Aufgabe, einen Block in der BAM (Block Availability Map, Verzeichnis der zur Verfügung stehenden Blöcke) als belegt zu kennzeichnen. Dies ist dann erforderlich, wenn wir im Direktzugriff Blöcke auf der Diskette beschrieben haben, die nicht Teil einer Datei sind und deshalb nicht automatisch als belegt gekennzeichnet sind. Werden dermaßen benutzte Blöcke nicht als belegt gekennzeichnt, können sie beim nächsten Schreiben in eine reguläre Datei überschrieben werden. Der Block-Allocate-Befehl hat folgende Syntax:

```
B-A Drive Track Sektor
```

Damit wird der entsprechende Block in der BAM als belegt gekennzeichnet und ist so vor dem Überschreiben durch andere Dateien geschützt. War der zu belegende Block bereits belegt so erhält man die Fehlermeldung 65,'NO BLOCK'.

```
100 OPEN 1,8,15
110 INPUT "TRACK, SECTOR ";T,S
120 PRINT# 1, "B-A 0";T;S
130 INPUT# 1, A$,B$,C$,D$
140 PRINT A$","B$","C$","D$
```

In dem kleinen Programm kann man Track und Sektor angeben, die man als belegt kennzeichnen will. War der Block noch frei, wird er belegt und die Meldung '00, OK,00,00' wird ausgegeben. War der Block jedoch bereits belegt, erscheint die Meldung '65,NO BLOCK,TT,SS'. Die Track und Sektornummer TT und SS geben jetzt den nächsten freien Block mit höherer Sektor und/ oder Tracknummer an. Erhält man diese Fehlermeldung, so weiß man, daß dieser Block belegt ist kann den nächsten freien Block benutzen. Erhält man bei der Fehlermeldung 65 jedoch als Track und Sektornummer jeweils eine Null zurück, so ist kein Block mit höherer Track und/ oder Sektornummer mehr frei. Das folgende Programm belegt automatisch den nächsten freien Sektor.

```
100 OPEN 1,8,15
110 INPUT "TRACK, SECTOR ";T,S
120 PRINT# 1, "B-A 0";T;S
130 INPUT# 1, A$,B$,TT,SS
140 IF A$ = "00" THEN 190
```

```
150 IF A$<>"65" THEN PRINT A$","B$","TT","SS : END
160 IF TT=0 THEN PRINT "KEIN FREIER BLOCK MEHR" : END
170 IF TT=18 THEN TT=19 : SS=0
180 T=TT : S=SS : GOTO 120
190 PRINT "TRACK" TT "SECTOR" SS "WURDE BELEGT"
```

Die Abfrage auf Track 18 in Zeile 170 verhindert, daß ein Block des Directorys belegt wird. Eine weitere Fehlermeldung in diesem Zusammenhang mit dem 'B-A'-Befehl ist noch interresant. Versucht man einen Block zu belegen, der gar nicht existiert, z.B. Track 20 Sektor 21, so erhält man die Fehlermeldung

```
66,ILLEGAL TRACK OR SECTOR,20,21
```

Die Kennzeichnung eines Blocks in der BAM als belegt verhindert das Überschreiben des Blocks durch andere Dateien. Der Block bleibt solange als belegt gekennzeichnet, bis der Befehl 'VALIDATE' ('COLLECT' in BASIC 4.0) auf die Diskette angewandt wird. Dieser Befehl konstruiert eine neue BAM. Dies geschieht folgendermaßen. Da sämtliche Blocks einer Datei mit einander verkettet sind, kann man so eine Datei über die Diskette verfolgen. Dies macht dem Validate-Befehl und markiert jeden Block, der zu einer Datei gehört, als belegt. Nicht geschlossene Dateien, im Directory mit '*' gekennzeichnet, werden dabei gelöscht. Dabei werden dann auch alle Blöcke, die mit 'B-A' belegt wurden und zu keiner regulären Datei gehören, wieder freigegeben. Hat man also im Direktzugriff Blöcke belegt, die nicht zu Dateien gehören, die im Directory erscheinen, so darf man den Validate-Befehl nicht anwenden, da sonst sämtliche Blöcke wieder freigegeben werden.

2.2.5 Der Block-Free-Befehl B-F

Der Block-Free-Befehl ist das Gegenstück zum Block-Allocate-Befehl und gibt einen Block in der BAM wieder frei. Die Syntax ist analog zum Block-Allocate-Befehl:

```
B-F Drive Track Sektor

100 OPEN 1,8,15
110 PRINT# 1, "B-F 0 20 9"
```

Mit diesem Befehl wird der Block in Track 20 Sektor 9 wieder in der BAM freigegeben. War der Block bereits freigegeben, so gibt es hier keine Fehlermeldung.

Das Belegen und Freigeben von Blocks haben nur Effekt auf das Überschreiben des Blocks mit regulären Dateien durch das DOS. Die Block-Write- und Block-Read-Befehle bleiben davon unbeeinträchtigt. Sie können mit diesen Befehlen sowohl belegte Blocks beschreiben, noch wird ein Block durch Beschreiben mit Block-Write in der BAM belegt. Haben Sie z.B. auf einer

2.3 Anwendungen des Direktzugriffs

Was läßt sich nun mit den Direktzugriffsbefehlen anfangen ?

Dazu kann man sich mehrere Anwendungen vorstellen. Die erste Möglichkeit besteht in der Manipulation einzelner Sektoren. Damit kann man eine Vielfalt von Aufgaben erfüllen. Es fängt an mit Manipulationen im BAM-Sektor, wo wir die Möglichkeit haben, den Diskettennamen oder die ID zu ändern. Dann bietet sich das Directory an. Dort könnten wir die ungenutzten Bytes für zusätzliche Informationen nutzen. Wir können Dateien einen anderen Namen geben und können die Verkettung der einzelnen Blocks einer Datei verfolgen und gegebenenfalls nach eigenen Vorstellungen ändern. Eine ganze Palette an Möglichkeiten tut sich auf, wenn es um den Filetyp der Datei geht. Wir können z.B. aus einer sequentiellen Datei eine Programmdatei machen, indem wir aus Filetyp 1 eine 2 machen. Wir können eine nichtgeschlossene Datei im Directory durch Setzen des Bit 7 schließen, aus $02 wird dann $82. Solche Dateien sind im Directory durch einen Stern gekennzeichnet; nach der obigen Änderung verschwindet der Stern. Eine vom DOS zwar berücksichtigte, per Befehl jedoch nicht erreichbare Eigenschaft einer Datei ist der Schutz vor dem Löschen. Dazu brauchen wir lediglich das Bit 6 des Filetyps setzen, z.B wird aus $82 dann $C2. Im Directory erscheint jetzt ein '<' hinter der Typbezeichnung. Die Datei ist nun gegen Scratchen immun. Damit können Sie z.B. wichtige Systemprogramme auf Ihrer Diskette gegen unbeabsichtigtes Löschen schützen. Diese und andere Möglichkeiten finden Sie in Kapitel 4.1.

Haben Sie derartige Manipulationen vor, so wäre es am komfortabelsten, wenn man sich einen kompletten Sektor von Diskette lesen könnte, ihn auf dem Bildschirm anzeigen, ändern und wieder auf Diskette schreiben könnte. Ein solches Programm, ein Disk-Monitor, ist in Kapitel 4.6 beschrieben. Ehe Sie jedoch mit solchen Experimenten beginnen, sollten Sie sich auf jeden Fall eine Kopie von Ihrer Diskette machen. Machen Sie nämlich gerade bei Directory und BAM Fehler, kann unter Umständen der ganze Disketteninhalt für Sie verloren sein.

Haben Sie schon mal aus Versehen eine Datei oder ein Programm auf Diskette gelöscht und sich dann darüber geärgert, daß Sie das komplette Programm neu eingeben mußten ? Falls Sie danach noch nicht auf die Diskette geschrieben haben, können Sie die Datei einfach zurückholen. Beim Löschen einer Datei wird nämlich lediglich im Directory der Filetyp auf 0 gesetzt und die belegten Blöcke in der BAM freigegeben. Sie brauchen jetzt nur der Directoryeintrag der Datei zu suchen und den Filetyp wieder einzusetzen: $81 für SEQ, $82 für PRG, $83 für USR und $84 für REL. Danach müssen Sie noch ein Validate machen, damit die Blöcke der Datei wieder als belegt gekennzeichnet werden, z.B. mit OPEN 1,8,15 : PRINT# 1, "V0".

Andere Anwendungen des Direktzugriff können z.B. dazu dienen,

eigene Dateistrukturen zu erzeugen, die das DOS nicht kennt. Sie müssen dann die Verwaltung der neuen Datei selbst übernehmen und benutzen zum Lesen und Schreiben die Direktzugriffsdatei. Eine solche Dateiform ist z.B. die ISAM-Datei. ISAM ist die Abkürzung für Index Sequentiell Acces Method, zu deutsch Index-sequentielle Zugrfiffsmethode heißt. Bei einer ISAM-Datei können Sie auf jeden Dateisatz direkt zugreifen, ähnlich wie bei einer realtiven Datei. Hierbei wird jedoch nicht über die Satznummer, sondern über einen sogenannten Zugriffsschlüssel oder Index zugegriffen. Dieser Index ist ein Feld des Datensatzes. Besteht ein Datensatz z.B. aus 5 Feldern, die Namen, Vornamen, Straße, Postleitzahl und Ort enthalten, so könnten wir den Namen als Zugriffsschlüssel definieren. Wollen wir nun den Datensatz des Kunden Müller lesen, so heißt der Befehl dazu einfach 'Lese Datensatz "Müller"'. Wir brauchen uns also nicht um irgendwelche Satznummern oder sonstige Ordnungskriterien kümmern und können im Klartext angeben, welchen Datensatz wir Lesen, Ändern, Schreiben oder Löschen wollen. In solchen ISAM-Dateisystemen ist meist der Index noch einmal separat abgespeichert zusammen mit den Informationen, wo der Datensatz auf Diskette zu finden ist. Eine solche ISAM-Datei-Verwaltung mit noch weitgehenderen Möglichkeiten, als sie hier beschrieben ist, finden Sie z.B. neben anderen Dingen im Programmentwicklungssystem MASTER, das auch für den Commodore 64 erhältlich ist.

2.4 Der Zugriff auf das DOS - Die Memory-Befehle

In Kapitel 2.2.6 haben wir bereits die Möglichkeit kennengelernt, Programme in den DOS-Speicher zu laden und dort auszuführen. Mit den Memory-Befehlen können wir nun auf jedes Byte des DOS zugreifen und Programme im RAM und ROM der Floppy ausführen. Wir können z.B. auf den Arbeitsspeicher des DOS zugreifen und z.B. die Anzahl der freien Blöcke auf der Diskette lesen oder den Diskettenname aus dem BAM-Puffer holen. Durch Schreiben in das DOS-RAM können wir Konstanten ändern, z.B. die Gerätenummer der Floppy oder die Anzahl der Leseversuche für einen Block, ehe eine Fehlermeldung gebracht wird. Weiterhin besteht die Möglichkeit, Routinen innerhalb des DOS-Speichers ausführen zu lassen. Das können sowohl Routinen des DOS als auch eigene Routinen sein, die in einem Pufferspeicher abgelegt und dort ausgeführt werden können, wie beim Block-Execute-Befehl. Voraussetzung für die erfolgreiche Nutzung dieser Befehle sind natürlich Kenntnise in 6502 Maschinensprache und in Arbeitsweise und Speicherbelegung des DOS; bei letzterem, so hoffen wir, kann Ihnen dieses Buch eine Hilfe sein. Es folgt nun eine Beschreibung der Befehle sowie Beispiele zu ihrer Anwendung.

2.4.1 Der Memory-Read-Befehl M-R

Mit diesem Befehl kann man jedes Byte des DOS lesen. Der Befehl wird über den Kommamdokanal übermittelt und stellt das gelesene Byte dann ebenfalls auf dem Kommamdokanal zur Verfügung, wo es mit GET# abgeholt werden kann. Die Syntax des Befehls sieht so aus:

```
M-R CHR$(LO) CHR$(HI)
```

Dabei bedeuten LO und HI das Low- und Highbyte der Adresse im DOS, die gelesen werden soll. Das folgende Programm fragt nach einer Adresse und liest den Inhalt dieser Adresse aus dem DOS.

```
100 INPUT "ADRESSE ";A
110 HI = INT (A/256)
120 LO = A-256*HI
130 OPEN 1,8,15
140 PRINT# 1, "M-R"; CHR$(LO); CHR$(HI)
150 GET# 1, A$
160 PRINT ASC(A$+CHR$(0))
```

Wollen wir z.B. die Anzahl der freien Blocks auf einer Diskette wissen, so brauchen wir nicht das komplette Inhaltsverzeichnis zu lesen, sondern können direkt die entsprechenden Bytes aus dem DOS-Speicher lesen. Dies kann z.B. dann nützlich sein, wenn man vom Programm aus Dateien anlegt und sich so vergewissern kann, ob noch genügend Platz auf der Diskette ist.

```
100 OPEN 1,8,15 ,"I0"
110 PRINT# 1, "M-R" CHR$(250) CHR$(2)
120 GET# 1, A$ : IF A$="" THEN A$=CHR$(0)
130 PRINT# 1, "M-R" CHR$(252) CHR$(2)
140 GET# 1, B$ : IF B$="" THEN B$=CHR$(0)
150 PRINT ASC(A$) + 256 * ASC (B$) "BLOCKS FREI"
160 CLOSE 1
```

Mit der angegebenen Syntax muß für jedes Byte, was gelesen werden soll, ein eigener 'M-R'-Befehl benutzt. Wie sich jedoch aus dem DOS-Listing entnehmen und durch Überprüfen bestätigen läßt, kann man auch mehrere aufeinander folgende Bytes mit einem 'M-R'-Befehl lesen. Man braucht nur die Anzahl der zu lesenden Bytes als dritten Parameter angeben:

```
M-R CHR$(LO) CHR$(HI) CHR$(ANZAHL)
```

Benutzen können wir dies z.B. dazu, um den Namen der Diskette aus dem BAM-pufferspeicher zu lesen. Dazu muß man wissen, daß die BAM beim Initialisieren oder sonst vor einem Dateizugriff in den Puffer ab Adresse $700 geladen wird, aus dem wir mit einem 'M-R'-Befehl den Namen der Diskette lesen können.

```
100 OPEN 1,8,15, "I0"
110 PRINT# 1, "M-R" CHR$(144) CHR$(7) CHR$(16)
120 INPUT# 1, A$
130 PRINT A$
```

Wir erhalten so auf einfache Weise den Namen der Diskette (16 Zeichen, aufgefüllt mit 'Shift Space'). Damit kann man vom Programm her überprüfen, ob die richtige Diskette eingelegt ist.

Auf diese Weise können auch die Diskettenpuffer gelesen werden, wenn man dem DOS auf die Spur kommen will. Ebenso besteht die Möglichkeit, Teile des DOS, die man nach eigenen Wünschen manipulieren will, vom ROM in einen Pufferspeicher zu kopieren, dort entsprechend zu ändern und dann zur Ausführung zu bringen. Doch dies gehört bereits in die beiden nächsten Abschnitte.

2.4.2 Der Memory-Write-Befehl M-W

Der gegensätzliche Befehl zum Memory-Read ist der Befehl zum Schreiben von Daten in den DOS-Speicher, Memory-Write, 'M-W'. Beschreiben läßt sich natürlich nur das DOS-RAM - Zeropage, Stack und Pufferspeicher sowie evtl. die Ein/Ausgabe-Bausteine. Hier ist von vorneherein an die Möglichkeit gedacht worden, mehrere aufeinder folgende Bytes mit einem Befehl zu schreiben. Die Syntax sieht so aus:

```
M-W CHR$(LO) CHR$(HI) CHR$(ANZAHL) CHR$(DATA1) CHR$(DATA2)
```

Dabei können soviele Daten übergeben werden, wie in Anzahl

spezifiziert ist, theoretisch also 255, da der Eingabepuffer jedoch nur 40 Zeichen faßt, ist die Anzahl auf 34 Bytes pro Befehl beschränkt. Eine mögliche Anwendung des Befehls dient zum Ändern der Gerätenummer der Floppy (siehe Programm 'DISK ADDR CHANGE', Kapitel 4.2.3). Die Adresse steht in zwei Speicherstellen in der Zeropage. In Adresse $77 gleich 119 steht die Gerätenummer plus $20 gleich 32 für LISTEN, also für den Empfang von Daten vom Computer. In der darauffolgenden Adresse steht die Gerätenummer plus $40 gleich 64 für TALK, also fürs Senden von Daten zum Computer. Da die Adressen separat gespeichert sind besteht also die Möglichkeit, für Senden und Empfangen verschiedene Adressen zu verwenden. Im folgenden Beispiel wird die Empfangsadresse auf 9 und die Sendeadresse auf 10 gesetzt.

```
100 OPEN 1,8,15
110 PRINT# 1, "M-W" CHR$(119) CHR$(0) CHR$(2)
                    CHR$(9+32) CHR$(10+64)
120 CLOSE 1
140 OPEN 1,9,15
150 OPEN 2,10,15
160 PRINT# 1,"I0"
170 INPUT# 2, A$,B$,C$,D$
180 PRINT A$ "," B$ "," C$ "," D$

00, OK,00,00
```

Programme können Sie so jedoch nicht laden, da hier ja der Dateinamen gesandt wird und unter der selben Adresse versucht wird, das Programm zu laden.

Das Ändern der Geräteadresse ist dann erforderlich, wenn Sie mehr als eine Floppy gemeinsam an einem Rechner betreiben wollen. Dazu ändert man die Geräteadressse der zweiten Floppy auf 9. Diese softwaremäßige Änderung bleibt jedoch nur solange erhalten, bis ein Reset (z.B. durch Ausschalten erfolgt). Soll die Änderung dauerhaft sein, kann dies im Gerät durch öffnen von Drahtbrücken geschehen.

Da viele Parameter des DOS im RAM stehen, können wir weitgehend die Funktion des DOS abändern, z.B. die Schrittweite, mit der die Sektoren in einem Track belegt werden (Adresse $69 gleich 105, enthält normalerweise 10). Ebenso können wir die Anzahl der Leseversuche bestimmen, ehe eine Fehlermeldung erzeugt wird (Adresse $6A gleich 106, Inhalt ist 5). Weitere Adressen von Parametern finden Sie in Kapitel 3.1.2.

2.4.3 Der Memory-Execute-Befehl M-E

Mit diesem Befehl nun können wir Maschinenprogramme im DOS-Speicher aufrufen und ausführen. Die Programme müssen mit RTS (Return from Subroutine, $60) abgeschlossen sein. Die Syntax des Befehls lautet

```
M-E CHR$(LO) CHR$(HI)
```

Dabei sind LO und HI wieder Low- und Highbyte der Startadresse der Maschinenroutine. Es besteht sowohl die Möglichkeit, Routinen des DOS-ROMs aufzurufen als auch eigene Routinen mit 'M-W' in einen Pufferspeicher zu schreiben und dort auszuführen. Als Beispiel dazu sehen wir einmal, wie man eine Routine aufrufen kann, die eine Fehlermeldung erzeugt. In Adresse $EFC9 steht z.B. der Aufruf zur Meldung 72, 'disk full'. Der Befehl sieht dann so aus:

```
100 OPEN 1,8,15
110 PRINT# 1, "M-E" CHR$(201) CHR$(239)
120 INPUT# 1, A$,B$,C$,D$
130 PRINT A$ "," B$ "," C$ "," D$
```

In Zeile 110 wird die Adresse $EFC9 in Lo-Byte $C9 gleich 201 und Hi-Byte $EF gleich 239 zerlegt und als Parameter des 'M-E'-Befehls gesandt. Dann wird der Fehlerkanal abgefragt und die Meldung ausgegeben.

```
72, DISK FULL,00,00
```

Will man eigene Programme in der Floppy ablaufen lassen, so wird man Sie in einen der Pufferspeicher schreiben und dort mit 'M-E' aufrufen. Soll dieses Programm öfter benutzt werden, so kann man den Inhalt des Puffers auf einem Block der Diskette speichern. Er kann dann später mit dem 'B-E'-Befehl ausgeführt werden, der den Inhalt des Blocks in den Puffer liest und dann die Routine automatisch startet. Als Anregung für eigene Programme im DOS können Sie ja einmal versuchen, das Directory in einer anderen Form auszugeben, die zusätzliche Parameter ähnlich wie im Programm in Kapitel 4.1.1. Zusätzlich könnte man noch die Anzahl der Dateien auf der Diskette zählen und mit ausgeben. Bei der Realisierung solch einer Routine können Sie sich am DOS-Listing orientieren, wie dort das Directory erzeugt wird. Ist man sich über das neue Format des Directorys im klaren, dürfte es keine Schwierigkeit mehr sein, die zusätzlichen Parameter, evtl. mit einer Überschrift, aus den Directoryeinträgen zu entnehmen und im gewünschten Format bereitzustellen.

2.4.4 Die User-Befehle U

Mit den User-Befehlen haben wir die zweite Möglichkeit, Programme in der Floppy auszuführen. Die User-Befehle haben folgende Syntax:

```
UX
```

Dabei kann X für einen Buchstaben von A bis J oder wahlweise eine Ziffer von 1 bis 9 und ':' (anstelle von 10) stehen. Beim Aufruf des Befehls wird zu folgenden Adressen im DOS gesprungen:

```
UA      U1      $CD5F   Ersatz für 'Block-Read'
UB      U2      $DC97   Ersatz für 'Block-Write'
UC      U3      $0500
UD      U4      $0503
UE      U5      $0506
UF      U6      $0509
UG      U7      $050C
UH      U8      $050F
UI      U9      $FF01
UJ      U:      $EAA0   Einschalt-Reset
```

Die Befehle U1 und U2 bzw. UA und UB kennen wir bereits; sie dienen als Ersatz für 'Block-Read' und 'Block-Write'. Die Befehle U3 bis U8 bzw. UC bis UH springen in den Puffer 2 ab Adresse $500 gleich 1280 (siehe Kapitel 2.1). Will man mehrere Befehle benutzen, kann dort eine Sprungtabelle auf die einzelnen Routinen stehen; wird bloß ein User-Befehl (U3) benutzt, kann das Programm direkt bei $500 beginnen.

Der User-Befehl UJ springt zum Resetvektor; damit wird die Floppy in den Einschaltzustand versetzt.

```
100 OPEN 1,8,15
110 PRINT# 1, "UJ"
120 FOR I=1 TO 1000 : NEXT
130 GET# 1, A$ : PRINT A$; :IF ST <> 64 THEN 130

73,CBM DOS V2.6 1541,00,00
```

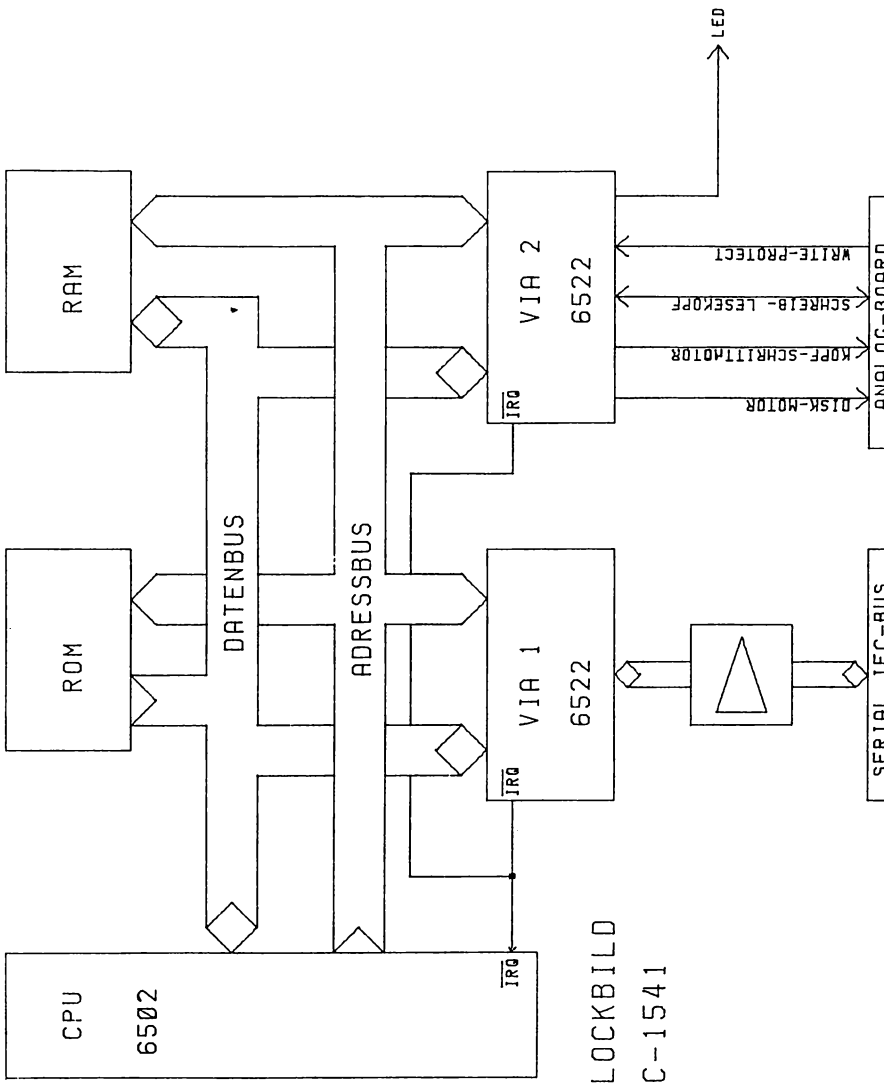
Zeile 120 wartet den Reset der Floppy ab. Dann wird in Zeile 130 die Einschaltmeldung der Floppy abgeholt.

Bei der Benutzung der User-Befehle können noch Parameter an die Routinen mit übergeben werden. Der komplette Befehlsstring wird im Eingabepuffer ab Adresse $200 gleich 512 abgelegt. Als Parameter wären z.B. Adressen, Befehlskodes und Dateinamen denkbar. Dadurch können die User-Befehle benutzt werden, um den Befehlsatz der Floppy zu erweitern oder um eigene Dateistrukturen zu verwirklichen. Sämtliche User-Befehle lassen sich durch 'M-E'-Befehle mit den entsprechenden Adressen ersetzen; der User-Aufruf ist doch kürzer und übersichtlicher.

Kapitel 3: Technik der Floppy und der Diskette

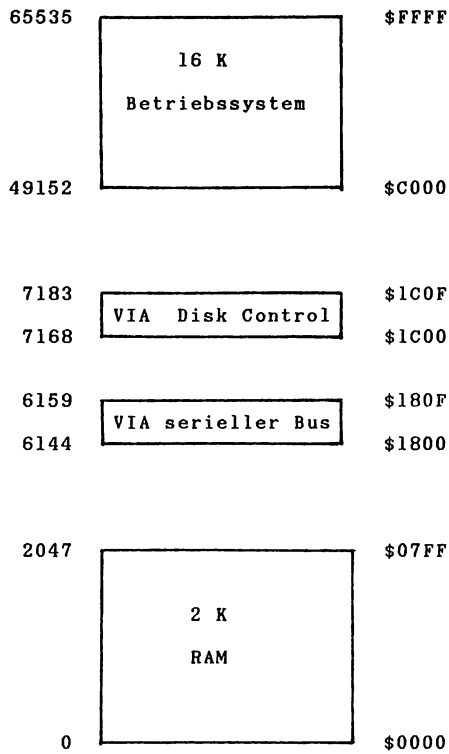
3.1 Der Aufbau der VC 1541

3.1.1 Blockschaltbild der Floppy



3.1.2 Memory-Map des DOS - ROM, RAM, I/O

Die Speicherbelegung der Floppy VC 1541



Die Belegung der I/O-Ports (VIA 6522)

VIA 6522 1, Port für seriellen Bus

```
$1800   Port B
$1801   Port A
$1802   Datenrichtung Port B
$1803   Datenrichtung Port A


PB 0:   DATA IN
PB 1:   DATA OUT
PB 2:   CLOCK IN
PB 3:   CLOCK OUT
PB 4:   ATN A
PB 5,6: Geräteadresse
CB 2:   ATN IN
```

VIA 6522 2, Port für Motor- und Schreib/ Lesekopfsteuerung

```
$1C00   Port B, Steuerport
$1C01   Port A, Daten vom und zum Schreib/ Lesekopf
$1C02   Datenrichtung Port B
$1C03   Datenrichtung Port A

PB 0:   STP I
PB 1:   STP 0    Schrittmotor für Kopfbewegung
PB 2:   MTR      Laufwerksmotor
PB 3:   ACT      LED am Laufwerk
PB 4:   WPS      Write Protect Switch


PB 7:   SYNC
CA 1:   Byte Ready
CA 2:   SOE
```

Die Belegung der wichtigsten Speicherstellen

0	$00	Befehlskode für Puffer 0
1	$01	Befehlskode für Puffer 1
2	$02	Befehlskode für Puffer 2
3	$03	Befehlskode für Puffer 3
4	$04	Befehlskode für Puffer 4 .
6	$06 - $07	Track und Sektor für Puffer 0
8	$08 - $09	Track und Sektor für Puffer 1
10	$0A - $0B	Track und Sektor für Puffer 2
12	$0C - $0D	Track und Sektor für Puffer 3
14	$0E - $0F	Track und Sektor für Puffer 4
18	$12 - $13	ID für Laufwerk 0
20	$14 - $15	ID für Laufwerk 1
22	$16 - $17	ID
32	$20 - $21	Flag für Kopftransport
48	$30 - $31	Pufferzeiger für Disk-Controller
57	$39	Konstante 8, Kennzeichen für Beginn Datenblockheader
58	$3A	Parity für Datenpuffer
61	$3D	Drivenummer für Disk Controller
63	$3F	Puffernummer für Disk Controller
67	$43	Anzahl der Sektoren pro Track bei der Formatierung
71	$47	Konstante 7, Kennzeichen für Beginn Datenblock
73	$49	Stackpointer
74	$4A	Schrittzähler für Kopftransport
81	$51	aktuelle Tracknummer bei der Formatierung
105	$69	Schrittweite bei Sektorzuteilung (10)
106	$6a	Anzahl der Leseversuche (5)
111	$6F - $70	Zeiger auf Adresse z.B für M- und B-Befehle
119	$77	Gerätenummer + $20 für Listen
120	$78	Gerätenummer + $40 für Talk
121	$79	Flag für Listen (1/0)
122	$7A	Flag für Talk (1/0)
124	$7C	Flag für ATN vom seriellen Bus empfangen
125	$7D	Flag für EOI vom seriellen Bus
127	$7F	Drivenummer
128	$80	Tracknummer
129	$81	Sektornummer
130	$82	Kanalnummer
131	$83	Sekundäradresse
132	$84	Sekundäradresse
133	$85	Datenbyte
139	$8B - $8D	Arbeitsspeicher für Division
148	$94 - $95	aktueller Pufferzeiger
153	$99 - $9A	Adresse Puffer 0 $300
155	$9B - $9C	Adresse Puffer 1 $400
157	$9D - $9E	Adresse Puffer 2 $500
159	$9F - $A0	Adresse Puffer 3 $600
161	$A1 - $A2	Adresse Puffer 4 $700
163	$A3 - $A4	Zeiger auf Eingabepuffer $200
165	$A5 - $A6	Zeiger auf Puffer für Fehlermeldung $2D5
181	$B5 - $BA	Record # lo, Blockzahl lo
187	$BB - $C0	Record # hi, Blockzahl hi
193	$C1 - $C6	Schreibzeiger für Rel-Datei
199	$C7 - $CC	Recordlänge für 'REL'-Dateien

212	$D4		Zeiger in Datensatz bei REL-Datei
213	$D5		Side Sektor Nummer
214	$D6		Zeiger auf Datenblock im Side-Sektor
215	$D7		Zeiger auf Datensatz in REL-Datei
231	$E7		Filetyp
249	$F9		Puffernummer
$100 -	$145	256-325	Stack
$200 -	$228	512-552	Puffer für Befehlsstring
$24A		586	Filetyp
$258		600	Recordlänge
$259		601	Track Side-Sektor
$25A		602	Sektor Side-Sektor
$274		628	Länge der Eingabezeile
$278		632	Zahl der Dateinamen
$297		663	Filebetriebsart
$280 -	$284	640-644	Track eines Files
$285 -	$289	645-649	Sektor eines Files
$2D5 -	$2F9	725-761	Puffer für Fehlermeldung
$2FA /	$2FC	762/764	Anzahl freie Blocks
$300 -	$3FF	768-1023	Puffer 0
$400 -	$4FF	1024-1279	Puffer 1
$500 -	$5FF	1280-1535	Puffer 2
$600 -	$6FF	1536-1791	Puffer 3
$700 -	$7FF	1792-2047	Puffer 4

3.2 Die Arbeitsweise des DOS - ein Überblick

Die VC 1541 ist ein intelligentes Diskettenlaufwerk mit eigenem Mikroprozessor und Betriebssystem (Disk Operating System, DOS). Dadurch wird kein Speicherplatz und keine Rechenzeit des angeschlossenen Rechners benötigt. Der Rechner braucht der Floppy lediglich Befehle zu übermitteln, die diese dann selbsttätig ausführt.

Die Floppy hat damit drei Aufgaben gleichzeitig zu erledigen: Zum Ersten muß sie den Datenverkehr vom und zum Rechner durchführen. Die zweite Aufgabe ist die Interpretation der Befehle und die Verwaltung von Dateien und den zugeordneten Übertragungskanälen und der Blockpuffer. Die dritte Aufgabe ist die hardwaremäßige Bedienung der Diskette; dazu gehört das Schreiben und Lesen einzelner Blocks auf der Diskette sowie das Formatieren von Disketten.

Diese Aufgaben muß bei der VC 1541 ein 6502-Mikroprozessor gleichzeitig durchführen. Dies ist nur mit Hilfe der Interrupttechnik möglich. Nur so können drei Programme quasi gleichzeitig ablaufen.

Das Hauptprogramm kümmert sich um die Interpretation und Ausführung der übermittelten Befehle. Das Empfangen von Daten und Befehlen vom Rechner wird nun per Interrupt erledigt. Will der Rechner ein Peripheriegerät ansprechen, so sendet er einen Impuls über die Leitung ATN (Attention, Achtung, siehe auch Kapitel 5.1). Damit löst er bei der Floppy einen Interrupt aus. Die Floppy unterbricht nun ihr laufendes Programm und merkt sich, daß der Rechner Daten senden wollte. Jetzt wird erst der ursprüngliche Befehl abgearbeitet. Danach kann die Floppy nun weitere Daten und Befehle vom Rechner annehmen und verarbeiten. Ist der Befehl abgearbeitet, so steht die Floppy in einer Warteschleife, bis neue Befehle vom Rechner kommen.

Das Abarbeiten der Befehle in dieser Ebene beschränkt sich jedoch auf die logische Verarbeitung der Befehle, die Verwaltung der Übertragungskanäle vom und zum Rechner sowie die Bereitstellung und Abholung der zu schreibenden bzw. zu lesenden Daten in die dafür vorgesehenen Pufferspeicher. Die Aufgaben eines 'Disk Controllers', das Formatieren von Disketten sowie das Schreiben und Lesen einzelner Blocks, müssen ebenfalls vom Prozessor ausgeführt werden.

Diese Aufgaben werden wieder interruptgesteuert durchgeführt. Durch einen eingebauten Zeitgeber ('Timer') wird ca. alle 14 Millisekunden das reguläre Programm der Floppy unterbrochen und in ein Programm verzweigt, das die Aufgaben eines Disk-Controllers erfüllt. Die Kommunikation zwischen den beiden eigenständigen Programmen geschickt über gemeinsam benutzte Speicherstellen, in die das Hauptprogramm Befehlskodes für das Disk-Controller-Programm ablegt. Wird nun das Interruptprogramm aktiv, so schaut es in diesen Speicher-

stellen nach, ob irgendwelche Aktivitäten verlangt werden. z.B. eine Diskette formatieren. Ist dies der Fall, so werden z.B. Laufwerks- und Kopfmotoren in Bewegung gesetzt. Nach Beenden der Interruptroutine schaut das Hauptprogramm wieder in bestimmten Speicherstellen nach, ob die Aufgabe vom Disk-Controller schon erledigt wurde oder ob noch weiter gewartet werden muß. Ebenso wird auf diese Weise dem Hauptprogramm mitgeteilt, ob irgendwelche Fehlerbedingungen, z.B. ein Read Error aufgetreten sind oder ob die Schreibschutzmarke geklebt war. Das Hauptprogramm kann dann entsprechend reagieren und z.B. eine Fehlermeldung bereitstellen.

Bei den großen CBM-Floppys wird als Disk-Controller ein eigener, zweiter Mikroprozessor von Typ 6504 eingesetzt. Die Kommunikation geschieht wieder über gemeinsame Speicherstellen.

Eine Übersicht über die Speicherbelegung des DOS sowie der Ein-Ausgabe Bausteine zur Bedienung von Diskette und seriellem Bus finden Sie im vorhergehenden Kapitel.

Diese Übersicht über die Arbeit des DOS kann natürlich nur einen groben Überblick geben. Wollen Sie sich genauer informieren, so können Sie das DOS-Listing der VC 1541 in Kapitel 3.5 zu Rate ziehen, in dem das komplette 16K-Betriebssystem ausführlich dokumentiert ist.

3.3 Der Aufbau der VC 1541-Diskette

Die Diskette der VC 1541 ist in 35 Spuren mit je 17 bis 21 Sektoren aufgeteilt. Die Gesamtzahl der Sektoren beträgt 683. Da das Directory die gesamte Spur 18 belegt, stehen 664 Datenblöcke zur Verfügung, die jeweils 256 Bytes aufnehmen können. Die Spuren sind wie folgt belegt:

SPUR	ANZAHL DER SEKTOREN
1 BIS 17	21
18 BIS 24	19
25 BIS 30	18
31 BIS 35	17

Die unterschiedliche Anzahl der Sektoren je Spur ist bedingt durch die Verkürzung der Spuren zum Mittelpunkt hin.

3.3.1 Die BAM der VC 1541

BAM ist die Abkürzung für Block-Availability-Map. Sie hat die Aufgabe, die Blöcke als belegt oder frei zu kennzeichnen. Nach jeder Manipulation der Blöcke (speichern, löschen, usw) wird die BAM aktualisiert. Wenn anhand der BAM festgestellt wird, daß ein zu speicherndes File mehr Blöcke benötigt, als verfügbar sind, so wird eine Fehlermeldung ausgegeben. Beim Eröffnen eines Files wird die BAM in den DOS-Speicher übernommen, parallel mit den Übertragungsbefehlen aktualisiert und beim Schließen der Datei zurück auf die Diskette geschrieben. Befehle, die Schreib- oder Löschfunktion haben, lesen die BAM, aktualisieren und schreiben sie wieder zurück. Die BAM ist auf Spur 18, Sektor 0 folgendermaßen organisiert:

Spur 18, Sektor 0

BYTE		INHALT	BEDEUTUNG
0,1	($00-$01):	$12,$01	Spur und Sektor des ersten Blocks der Directory
2	($02)	$41	ASCII-Zeichen "A"; zeigt 1541-Format an
3	($03)	$00	Null-Flag für zukünftige Benutzung
4-143	($04-$8F):		Bitmuster der belegten bzw. nicht belegten Blöcke *

*) 1 = Block nicht belegt ; 0 = Block belegt

Das Bitmuster der Blöcke ist so organisiert, daß jeweils 4 Bytes eine Spur kennzeichnen. Wie es der folgenden Tabelle zu entnehmen ist, enthält das erste der 4 Bytes die Anzahl der freien Blöcke dieser Spur. Die restlichen 3 Bytes (24 Bits) kennzeichnen die freien oder belegten Blöcke dieser Spur.

Struktur des BAM-Eintrags einer Spur:

BYTE	INHALT
0	Zahl der verfügbaren Blöcke der Spur
1	Bitmuster der Sektoren 0-7
2	Bitmuster der Sektoren 8-15
3	Bitmuster der Sektoren 16-23

4 Bytes einer Spurkennzeichnung in der BAM:

Spur 18, Sektor 0, Byte 4-7 (Spur 1)			
00001010	00000000	00000011	11111111
($0A)	($00)	($00)	($3F)
10 freie Blöcke	1 = nicht belegt 0 = belegt		

Durch Programmierung einer Schleife, die das jeweils 1. Byte liest und aufaddiert, ist es möglich, die freien Blöcke der gesamten Diskette zu ermitteln.

3.3.2 Das Directory

Das Directory ist das "Inhaltsverzeichns" der Diskette. Sie enthält folgende Informationen:

- Diskettenname
- ID der Diskette
- Nummer der DOS-Version
- Filenamen
- Filetypen
- Blocks pro File
- freie Blöcke

Dieses Directory wird mit dem Befehl 'LOAD "$",8' in den Speicher geladen. Dabei wird ein evtl. gespeichertes Programm zerstört! Mit dem Befehl 'LIST' kann sie dann auf dem Bildschirm ausgegeben werden.
Das Directory belegt die gesamte Spur 18 der Diskette. Dem Vorspann der Directory folgen die Fileeinträge. Jeder Block nimmt maximal 8 Fileeinträge auf. Da die BAM und der Vorspann der Directory 1 Block belegen, stehen auf dieser Spur noch 18

Blöcke für Fileeinträge zur Verfügung. Es können demnach auf einer Diskette maximal 144 Files (18 Blöcke mit je 8 Einträge) verwaltet werden.

Format des Vorspanns des Directorys:

Spur 18, Sektor 0		
BYTE	INHALT	BEDEUTUNG
144-161 ($90-$A1):		Name der Diskette (ergänzt mit "SHIFT SPACE")
162,163 ($A2,$A3):		ID-Kennzeichnung der Disk
164 ($A4)	$A0	"SHIFT SPACE"
165,166 ($A5,$A6):	$32,$41	ASCII-Zeichen "2A" (Format)
167-170 ($A7-$AA):	$A0	"SHIFT SPACE"
171-255 ($AB-$FF):	$00	wird nicht benutzt, ist mit Nullen ausgefüllt *
* Die Bytes 180 bis 191 können auf manchen Disketten den Inhalt "BLOCKS FREE" haben		

Der Name der Diskette

Bei der Formatierung wird der Diskettenname, der aus maximal 16 Zeichen besteht, festgelegt. Werden weniger als 16 Zeichen angegeben, so wird der Rest mit "SHIFT SPACE" ($A0) ausgefüllt. Die folgende BASIC-Routine liest den Namen und speichert ihn in eine String-Variable:

```
100 OPEN 15,8,15,"I0"       :REM BEFEHLSKANAL 15 ÖFFNEN UND
                             NEU INITIALISIEREN
110 OPEN 2,8,2,"#"          :REM DATENKANAL 2 ÖFFNEN
120 PRINT#15,"B-R";2;0;18;0:REM SPUR 18, BLOCK 0 LESEN
                             UND IN KANAL 2 ABLEGEN
130 PRINT#15,"B-P";2;144    :REM BUFFER-POINTER AUF BYTE 144
140 DN$=""                  :REM STRING DN$ LÖSCHEN
150 REM SCHLEIFE ZUM EINLESEN DER 16 BYTES DES NAMENS
160 ::FOR I=1 TO 16
170 ::GET#2,X$              :REM LESEN EINES BYTES
180 ::IF ASC(X$)=160THEN200:REM SHIFT-SPACE NICHT ÜBERNEHMEN
190 ::DN$=DN$+X$            :REM BYTE AN DN$ ANHÄNGEN
200 NEXT I
210 CLOSE 2:CLOSE 15        :REM KANÄLE SCHLIESSEN
```

Nach Ablauf dieser Routine steht der Diskettenname in dem String DN$ zur Verfügung. Diese Routine kann z.B. in Anwendungsprogrammen sinnvoll sein, um festzustellen, ob die richtige Diskette eingelegt ist.

ID-Kennzeichnung der Diskette

Die Disketten-ID besteht aus 2 Zeichen und wird beim Formatieren der Diskette bestimmt. Anhand dieser Kennzeichnung stellt das DOS den Wechsel der Diskette fest, was zum Initialisieren der neuen Diskette notwendig ist. Als Initialisieren bezeichnet das Einlesen der BAM in den Speicher des Laufwerkes. Damit das DOS stets die aktuelle BAM im Speicher vorfindet, sollte die ID beim Formatieren immer unterschiedlich sein. Sollte dies nicht der Fall sein, so muß nach einem Diskettenwechsel mit dem Befehl INITIALIZE "von Hand" initialisiert werden.

3.3.3 Das Format des Directorys

Die Blöcke 1 bis 19 der Spur 18 beinhalten die Einträge der Files. Die ersten beiden Bytes eines Blocks zeigen auf den Block mit den nächsten Fileeinträgen. Sollte kein weiterer Block folgen, so beinhalten diese beiden Bytes $00 und $FF.

Spur 18, Sektor 1		
Byte		Inhalt
0,1	($00,$01)	Spur und Sektor des nächsten Blocks der Directory
2-31	($02-$1F)	Eintrag des 1. Files
34-63	($22-$3F)	Eintrag des 2. Files
66-95	($42-$5F)	Eintrag des 3. Files
98-127	($62-$7F)	Eintrag des 4. Files
130-159	($82-$9F)	Eintrag des 5. Files
162-191	($A2-$BF)	Eintrag des 6. Files
194-223	($C2-$DF)	Eintrag des 7. Files
226-255	($E2-$FF)	Eintrag des 8. Files

Format eines Directory-Eintrags

Jeder Fileeintrag besteht aus 30 Bytes, deren Funktion im Folgendem beschrieben sind:

BYTE		INHALT
0	($00)	Filetyp
1,2	($01,$02)	Spur und Sektor des ersten Datenblocks
3-18	($03-$12)	Filename (ergänzt mit "SHIFT SPACE")
19,20	($13,$14)	Nur bei relativen Files benutzt (Spur und Sektor des ersten Side-Sector-Blocks)
21	($15)	Nur bei relativen Files benutzt (Recordlänge)
22-25	($16-$19)	Nicht benutzt
26,27	($1A-$1B)	Spur und Sektor des neuen Files beim Überschreiben mit dem Klammeraffen
28,29	($1C-$1D)	Anzahl der Blocks im File (Low-Byte, High-Byte)

Kennzeichnung des Filetyps

Das Byte 0 des Fileeintrags kennzeichnet den Filetyp. Zur Kodierung der 5 Filetypen werden die Bits 0-2 benutzt. Das Bit 7 kennzeichnet, ob das File ordnungsgemäß geschlossen ist. Wird ein File geöffnet, so wird der entsprechende Filetyp gesetzt. Beim Schließen dieses Files wird dann das Bit 7 gesetzt. Ein nicht geschlossenes File wird im aufgelisteten Directory mit einem Stern vor dem Filetyp gekennzeichnet. Wird z.B. Ein sequentielles File "TEST" geöffnet und anschließend das Directory aufgelistet, so wird dieses File so im Directory dargestellt:

```
12    "TEST"           *SEQ
```

Wird das File wieder geschlossen, so erscheint der Stern bei nochmaligem Auflisten des Directorys nicht mehr. Wird dieses File nicht geschlossen und später nochmals eröffnet, so erschient die Fehlermeldung "WRITE FILE OPEN".

Der Filetyp

Um die Funktion des Byte 0 im Fileeintrag, also den Filetyp, richtig zu verstehen, folgt nun eine Tabelle aller Filetypen:

Filetyp	Bitmaske geöffnet 7654 3210	HEX	Bitmaske geschlossen 7654 3210	HEX
DELeted	0000 0000	$00	1000 0000	$80
SEQuential	0000 0001	$01	1000 0001	$81
PRoGram	0000 0010	$02	1000 0010	$82
USeR	0000 0011	$03	1000 0011	$83
RELative	0000 0100	$04	1000 0100	$84

Vieleicht haben Sie erkannt, daß die Bits 3 bis 6 ohne Funktion sind. Als wir dies mit Hilfe des DOS-Listings nachprüften, stellten wir fest, daß das Bit 6 doch eine Funktion hat:

DAS BIT 6 DES FILETYPS KENNZEICHNET EIN GESCHÜTZTES FILE !

Setzt man dieses Bit auf 1, so kann das entsprechende File nicht mehr gelöscht werden. Dies wird im aufgelisteten Directory mit dem Zeichen '<' hinter dem Filetypen gekennzeichnet.
Da das Setzen dieses Bits eine Folge von komplizierten Befehlen erfordert, finden Sie in Kapitel 4 dieses Buches ein Programm, mit dem Sie Files schützen, freigeben und löschen können.

Spur und Sektor des ersten Datenblocks

Die Bytes 1 und 2 des Fileeintrags weisen auf den ersten Datenblock des Files. Dabei ist im Byte 1 die Spur und in Byte 2 der Sektor dieses Blocks enthalten. Dieser erste Datenblock enthält dann in den ersten beiden Bytes die Adresse des zweiten Datenblocks, usw. Um den letzten Datenblock zu identifizieren, enthält dieser den Wert $00 im ersten Byte. Das zweite Byte enthält die Anzahl der Bytes, die in diesem Block vom File belegt sind.
Diese Verkettung läßt sich mit Hilfe des DOS-MONITORS, der in diesem Buch enthalten ist, gut verdeutlichen.:

```
>:B0  A0 A0 A0 A0 A0 00 00 00        ...
>:B8  00 00 00 00 00 00 0B 00  ........
>:C0  00 00 81 13 09 54 31 32  .....T12
>:C8  2F 53 30 31 A0 A0 A0 A0  /S01
>:D0  A0 A0 A0 A0 A0 00 00 00        ...
>:D8  00 00 00 00 00 00 06 00  ........     Filetyp
>:E0  00 00 82 10 00 44 49 53  .....DIS     Spur 1. Block
>:E8  4B 20 41 44 44 52 20 43  K ADDR C     Sektor 1. Block
>:F0  48 41 4E 47 45 00 00 00  HANGE...
>:F8  00 00 00 00 00 00 04 00  ........     Anzahl Blöcke
```

Dies ist ein Auszug aus dem Directory (Spur 18, Sektor 1) der TEST/DEMO-Diskette. Verfolgen wir nun die Organisation des Files DISK ADDR CHANGE. Der Eintrag dieses Files beginnt bei Byte $E2 und endet mit Byte $FF. Dies ist ein PRG-File, was an dem Filetyp $82 in Byte $E2 zu erkennen ist. Dieses File umfasst 4 Blöcke auf der Diskette. Dies ist an den Bytes $FE und $FF ersichtlich. Die Bytes $E3 und $E4 des Eintrags adressieren den ersten Datenblock des Files ($10, $00, entspricht Spur 16, Sektor 0).
Schauen wir uns nun den Ausschnitt dieses Blocks einmal an:

```
>:00  10 0A 01 04 0F 04 64 00  ......$.
>:08  97 35 39 34 36 38 2C 31  .59468,1      Adresse 2. Block
>:10  32 00 39 04 6E 00 99 22  2.9....."
>:18  93 13 11 11 11 11 44 52  ......DR
>:20  49 56 45 20 41 44 44 52  IVE ADDR
>:28  45 53 53 20 43 48 41 4E  ESS CHAN
>:30  47 45 20 50 52 4F 47 52  GE PROGR
>:38  41 4D 22 00 59 04 6F 00  AM".Y./.
>:40  99 22 11 54 55 52 4E 20  ."·TURN
>:48  4F 46 46 20 41 4C 4C 20  OFF ALL
```

Dieser Block enthält den ersten Teil des Programms, das in der linken Charakter-Darstellung schwer zu lesen ist. Das liegt daran, daß BASIC-Programme auf Diskette genauso abgelegt werden, wie im Speicher des Rechners. Die BASIC-Befehle werden in Form eines Ein-Byte-Codes (Tokens genannt) abgekürzt. Somit ist nur der Text zu erkennen. Die ersten beiden Bytes dieses Datenblocks weisen nun auf den zweiten Datenblock ($10 und $0A, also Spur 16, Sektor 10), dessen Auschnitt nun folgt:

```
>:00  10 14 34 30 00 1D 05 A0  ..40...
>:08  00 8D 20 33 30 30 3A 20  .. 300:       Adresse 3. Block
>:10  8F 20 46 49 4E 44 20 44  . FIND D
>:18  52 49 56 45 20 54 59 50  RIVE TYP
>:20  45 00 39 05 AA 00 8D 20  E.9. ..
>:28  36 30 30 3A 20 8F 20 43  600: . C
>:30  48 41 4E 47 45 20 41 44  HANGE AD
>:38  44 52 45 53 53 00 68 05  DRESS.(.
>:40  B4 00 99 22 11 54 48 45  ..".THE
>:48  20 53 45 4C 45 43 54 45   SELECTE
```

Das Programm wird in diesem Block fortgesetzt. Die Bytes $00 und $01 zeigen nun auf den 3. Datenblock des Files ($10, $14, Spur 16, Sektor 20):

```
>:00  10 08 31 30 30 30 00 23  ..1000.#
>:08  06 54 01 8B 20 43 B2 32  .T.. C 2      Adresse 4. Block
>:10  35 34 20 A7 20 4D 54 B2  54  MT
>:18  31 31 39 3A 20 8F 3A 20  119: .:
>:20  32 30 33 31 20 56 32 2E  2031 V2.
>:28  36 00 45 06 5E 01 8B 20  6.E. ..
>:30  43 B2 32 32 36 20 A7 20  C 226
>:38  4D 54 B2 35 30 3A 20 8F  MT 50: .
>:40  3A 20 32 30 34 30 20 56  : 2040 V
>:48  31 2E 32 00 67 06 68 01  1.2. .(.
```

Dies ist der vorletzte Block des Programms. Sie haben sicher erkannt, daß die Datenblöcke zwar in der gleichen Spur, jedoch nicht nacheinander angeordnet sind. Das heißt aber nicht, daß die Belegung der Blöcke ohne Sytem erfolgt. Der erste Datenblock ist der Block 0. Der nächste ist der Block 10, also 10 Blöcke weiter. Es werden immer 9 Blöcke übersprungen, was sich im weiteren Verlauf bewahrheitet. Der 3. Datenblock ist der Block 20. Das DOS fängt wieder beim ersten Block an, wenn der errechnete Block den höchsten Block

überschreitet. Weil sich die Spur 16 über 21 Blöcke erstreckt, ist der letzte Datenblock der Block 8.Die ersten beiden Bytes dieses 3. Blocks adressieren Ihn:

```
>:00  00 F8 5A 42 B2 31 20 A7  . ZB 1
>:08  20 34 34 30 00 14 07 AE   440...
>:10  01 8B 20 53 54 20 A7 20  .. ST
>:18  31 30 30 30 00 45 07 B8  1000.E.
>:20  01 98 31 35 2C 22 4D 2D  ..15,"M-
>:28  52 22 C7 28 31 37 32 29  R" (172)
>:30  C7 28 31 36 29 3A A1 23   (16): #
>:38  31 35 2C 5A 43 24 3A 5A  15,ZC$:Z
>:40  43 B2 C6 28 5A 43 24 AA  C f(ZC$
>:48  C7 28 30 29 29 00 66 07  g(0)).&.
```

Zeichen letzter Block

Anzahl belegter Bytes dieses Blocks

Hier ist das Ende des Programms durch den Wert $00 im Byte $00 gekennzeichnet. Das Byte $01 gibt die Anzahl der Bytes an, die von dem Programm in diesem letzten Block belegt sind ($F8 enstpricht 248 Bytes). Nun läßt sich leicht die Größe des Programms ermitteln:

```
3 Blöcke mit je 254 Bytes = 762 Bytes
letzter Block             = 248 Bytes
                            ----------
Größe des Programms:        1100 Bytes
                            ==========
```

Der Filename

Der Filename ist in den Bytes 3-18 des Fileeintrags enthalten. Er umfasst maximal 16 Zeichen. Sollte der Name kleiner als 16 Zeichen sein, so wird der Rest wie beim Diskettennamen mit "SHIFT SPACE" ($A0) ausgefüllt.

Spur und Sektor des neuen Files beim "Überschreiben"

Wird ein File durch Angabe des Klammeraffens vor dem Filenamen überschrieben, so wird das neue File zuerst komplett abgespeichert. Es wird aber für dieses File kein Eintrag erstellt, weil das File ja bereits unter diesem Namen existiert. Die Adresse des ersten Blocks des neuen Files wird in den Bytes 26 und 27 des Eintrags gespeichert. Ist das neue Programm abgelegt, wird das alte gelöscht, indem lediglich die bisher von diesem File belegten Blöcke in der BAM als frei gekennzeichnet werden. Nun wird die Adresse des ersten Datenblocks des neuen Files in die Bytes 1 und 2, Adresse des ersten Datenblocks des Files, gebracht und das File ist "überschrieben"

Anzahl der Blöcke im File

In den beiden Bytes 28 und 29 des Fileeintrags ist die Länge des Files in Blöcken angegeben. Eine Datei umfaßt mindestens einen und höchstens 644 Blöcke. Das erste Byte ist das Low-Byte, d.h. der rechte Teil der 2-Byte-Zahl. Das zweite Byte ist das High-Byte. Haben Sie z.B. mit dem DISK-MONITOR die Filelänge $1F,$00 ermittelt, so umfasst das File 31 Blöcke.

3.4 Die Organisation von relativen Dateien

Relative Dateien unterscheiden sich von gewöhnlichen sequentiellen Dateien dadurch, daß hier auf jeden Datensatz direkt zugegriffen werden kann. Deshalb muß hier außer den Daten selbst noch zusätzlich eine Datei abgespeichert werden, in der steht, wo jeder Datensatz zu finden ist.

Diese Aufgabe wird von der Floppy automatisch ohne Ihr Zutun erledigt. Sehen wir uns die Organisation der relativen Datei nun einmal etwas näher an.

Dazu öffnen wir eine relative Datei mit einer Datensatzlänge von 100:

```
OPEN 2,8,2, "REL-DATEI,L,"+CHR$(100)
```

und legen den Datensatz Nr. 70 an.

```
OPEN 1,8,15
PRINT#1, "P"+CHR$(2)+CHR$(70)+CHR$(0)+CHR$(1)
PRINT#2, "DATENSATZ 70"
CLOSE 2 : CLOSE 1
```

Der Directoryeintrag sieht dann so aus:

```
>:00  .. .. 84 11 00 52 45 4C   ...REL
>:08  2D 44 41 54 45 49 A0 A0 -DATEI
>:10  A0 A0 A0 A0 A0 11 0A 64       ..$
>:18  00 00 00 00 00 00 1D 00 ........
```

Das erste Byte $84 kennzeichnet eine relative Datei. Die nächsten beiden Byte kennzeichnen den ersten Track und Sektor der eigentlichen Daten ($11, $00; Track 17 Sektor 0); genau wie bei einer sequentiellen Datei. Es folgt wieder wie üblich der Name der Datei (16 Zeichen, aufgefüllt mit 'Shift Space', $A0). Jetzt folgen drei Einträge, die wir bei sequentiellen Dateien nicht kennen. Die ersten beiden Byte weisen auf Track und Sektor des ersten sogenannten Side-Sektor-Blocks, der die Zeiger auf jeden Datensatz enthält und den wir gleich näher kennenlernen werden ($11, $0A; Track 17 Sektor 10). Das nächste Byte enthält die Datensatzlänge, ein Wert zwischen 1 und 254, in unserem Falle $64 gleich 100. Die Annehmlichkeit, auf jeden Datensatz direkt zugreifen zu können, erfordert eine feste Länge für jeden Datensatz, die wir beim Anlegen der relativen Datei definieren müssen. Die restlichen Bytes im Directoryeintrag haben wieder die übliche Bedeutung; so enthalten die beiden letzten Bytes wieder die Anzahl der Blocks, die durch die Datei belegt werden (lo- und hi-Byte, $1D und $00 gleich 29).

Wie sieht nun so ein Side-Sektor-Block aus und welche Aufgabe hat er ?

Die Side-Sektor-Blocks enthalten die Track- und Sektor-Zeiger auf die einzelnen Datenblocks. Wollen wir zum Beispiel den 70. Datensatz aus unserer relativen Datei lesen, so schaut die Floppy im Side-Sektor-Block nach, auf welchem Track und Sektor der Datensatz steht und kann dann direkt diesen Block lesen. Dadurch wird verhindert, daß die gesamte Datei bis zum 70. Satz gelesen werden muß. Es müssen also nur zwei Blocks gelesen werden, um den Datensatz zu erhalten. Nach dieser etwas vereinfachten Darstellung sehen wir und jetzt den genauen Aufbau eines Side-Sektor-Blocks an. Wir beziehen uns wieder auf die oben geöffnete Datei.

```
>:00  00 47 00 64 11 0A 00 00 .G.$....
>:08  00 00 00 00 00 00 00 00 ........
>:10  11 00 11 0B 11 01 11 0C ........
>:18  11 02 11 0D 11 03 11 0E ........
>:20  11 04 11 0F 11 05 11 10 ........
>:28  11 06 11 11 11 07 11 12 ........
>:30  11 08 11 13 11 09 11 14 ........
>:38  10 08 10 12 10 06 10 10 ........
>:40  10 04 10 0E 10 02 10 0C ........
>:48  00 00 00 00 00 00 00 00 ........
>:50  00 00 00 00 00 00 00 00 ........
>:58  00 00 00 00 00 00 00 00 ........
>:60  00 00 00 00 00 00 00 00 ........
>:68  00 00 00 00 00 00 00 00 ........
>:70  00 00 00 00 00 00 00 00 ........
>:78  00 00 00 00 00 00 00 00 ........
>:80  00 00 00 00 00 00 00 00 ........
>:88  00 00 00 00 00 00 00 00 ........
>:90  00 00 00 00 00 00 00 00 ........
>:98  00 00 00 00 00 00 00 00 ........
>:A0  00 00 00 00 00 00 00 00 ........
>:A8  00 00 00 00 00 00 00 00 ........
>:B0  00 00 00 00 00 00 00 00 ........
>:B8  00 00 00 00 00 00 00 00 ........
>:C0  00 00 00 00 00 00 00 00 ........
>:C8  00 00 00 00 00 00 00 00 ........
>:D0  00 00 00 00 00 00 00 00 ........
>:D8  00 00 00 00 00 00 00 00 ........
>:E0  00 00 00 00 00 00 00 00 ........
>:E8  00 00 00 00 00 00 00 00 ........
>:F0  00 00 00 00 00 00 00 00 ........
>:F8  00 00 00 00 00 00 00 00 ........
```

Die ersten beiden Byte zeigen wie üblich auf Track und Sektor des nächsten Side-Sektor-Blocks. (In unserem Beispiel existiert kein weiterer Side-Sektor-Block und es werden nur $47 = 71 Bytes genutzt.) Byte 2 enthält die Nummer des Side-Sektor-Blocks, 0. Dazu muß man wissen, daß eine relative Datei maximal 6 solcher Blocks enthalten kann; die Nummerierung geht von 0 bis 5. In Byte 3 steht die Datensatzlänge $64 = 100. Die nächsten zwölf Byte (Nummer 4 bis 15) enthalten jeweils Track- und Sektor-Zeiger auf die 6 Side-Sektor-Blöcks (0,0 falls der Block noch nicht angelegt ist). Ab Byte 16 stehen die eigentlichen Zeiger auf die

Daten, und zwar die Track- und Sektor-Zeiger auf die ersten 120 Datenblöcke (in unserem Falle nur 28 Zeiger). Soll nun ein bestimmter Datensatz gesucht werden, so kann das DOS aus der Datensatznummer und der Datensatzlänge genau berechnen, auf welchem Block die Daten stehen und ab welcher Position innerhalb des Blocks der Datensatz beginnt. Nehmen wir dazu folgendes Beispiel:

Wir wollen den 70. Datensatz aus unserer Datei mit einer Datensatzlänge mit 100 Zeichen lesen. Wir haben dann folgende Rechnung durchzuführen:

```
(70-1) * 100 / 254
```

Wir erhalten als Ergebnis 27 und einen Rest von 42. Das DOS weiß nun, daß der Datensatz im 27. Datenblock ab der Position 42+2 gleich 44 zu finden ist. Die Rechnung erklärt sich folgendermaßen: Jeder Block enthält 256 Byte, von denen die ersten beiden Bytes als Zeiger auf den nächsten Block gebraucht werden, es bleiben also 254 Bytes zur Datenspeicherung übrig. Aus Datensatznummer und Datensatzlänge haben wir die Bytenummer innerhalb der Datei berechnet. Wenn wir diesen Wert durch die Anzahl der Bytes pro Block dividieren, erhalten wir die Nummer des Blocks, in dem der Datensatz steht, während der Rest der Division die Position innerhalb des Blocks ergibt (plus 2, da die ersten beiden Byte als Zeiger dienen). Geht der Datensatz über das Blockende hinaus, so muß auch der nächste Datensatz gelesen werden.

In unserem Beispiel steht der 27. Datenblock in Track $10 = 16 und Sektor $0C = 12. Wenn wir diesen Block lesen, erhalten wir folgendes Bild:

```
>:00  00 F3 00 00 00 00 00 00 ........
>:08  00 00 00 00 00 00 00 00 ........
>:10  00 00 00 00 00 00 00 00 ........
>:18  00 00 00 00 00 00 00 00 ........
>:20  00 00 00 00 00 00 00 00 ........
>:28  00 00 00 00 44 41 54 45 ....DATE
>:30  4E 53 41 54 5A 20 37 30 NSATZ 70
>:38  0D 00 00 00 00 00 00 00 ........
>:40  00 00 00 00 00 00 00 00 ........
>:48  00 00 00 00 00 00 00 00 ........
>:50  00 00 00 00 00 00 00 00 ........
>:58  00 00 00 00 00 00 00 00 ........
>:60  00 00 00 00 00 00 00 00 ........
>:68  00 00 00 00 00 00 00 00 ........
>:70  00 00 00 00 00 00 00 00 ........
>:78  00 00 00 00 00 00 00 00 ........
>:80  00 00 00 00 00 00 00 00 ........
>:88  00 00 00 00 00 00 00 00 ........
>:90  FF 00 00 00 00 00 00 00 ........
>:98  00 00 00 00 00 00 00 00 ........
>:A0  00 00 00 00 00 00 00 00 ........
>:A8  00 00 00 00 00 00 00 00 ........
```

```
>:B0  00 00 00 00 00 00 00 00 ........
>:B8  00 00 00 00 00 00 00 00 ........
>:C0  00 00 00 00 00 00 00 00 ........
>:C8  00 00 00 00 00 00 00 00 ........
>:D0  00 00 00 00 00 00 00 00 ........
>:D8  00 00 00 00 00 00 00 00 ........
>:E0  00 00 00 00 00 00 00 00 ........
>:E8  00 00 00 00 00 00 00 00 ........
>:F0  00 00 00 00 FF 00 00 00 ........
>:F8  00 00 00 00 00 00 00 00 ........
```

Erhalten wir bei der Berechnung eine Blocknummer über 120, so befindet sich der Zeiger auf den Datensatz nicht mehr im ersten Side-Sektor-Block, sondern in einem der nächsten Blöcke. Hier können Sie wieder die Blockzahl durch 120 dividieren und Sie erhalten die Nummer des Side-Sektor-Blocks. Der Rest gibt dann wieder die Nummer des Datenzeigers innerhalb dieses Blocks an. Haben wir als Blocknummer z.B. 425 erhalten, so erhalten wir 3 Rest 65. Wir müssen also Side-Sektorblock 3 lesen und dort den Zeiger auf den 65. Datenblock holen. Da jeder Side-Sektor-Block die Track- und Sektornummern der anderen Side-Sektor-Blocks enthält, ist wiederum nur ein weiterer Lesezugriff erforderlich. Für den Zugriff auf einen Datensatz einer relativen Datei sind also zwischen 2 und 4 Blockzugriffe erforderlich. Da die eigentlichen Datensätze einer relativen Datei genau wie bei einer sequentiellen Datei mit einander verkettet sind, ist auch ein sequentielles Lesen oder Schreiben ohne Angabe einer Datensatznummer möglich. Dabei wird nach jedem Schreiben oder Lesen der Zeiger auf den jeweils nächsten Datensatz gesetzt.

Beim Anlegen und Erweitern einer relativen Datei geschieht folgendes:
Zuerst wird ein Directoryeintrag für die relative Datei erzeugt, der den Eintrag über die beim Öffnen angegebene Länge enthält. Gleichzeitig werden zwei Datenkanäle für die relative Datei reserviert (einer für die Daten selbst, der andere für die Side-Sektor-Blöcke). Wird jetzt der Recordzeiger auf einen bestimmten Datensatz gesetzt, wird erst geprüft, ob dieser Datensatz schon existiert. Ist dies der Fall, werden die entsprechenden Blocks gelesen und die Zeiger auf diesen Datensatz gesetzt, der nun gelesen oder geschrieben werden kann. Existierte dieser Datensatz noch nicht, so wird er angelegt. Dabei werden auch alle evtl. noch nicht existierende Datensätze mit kleinerer Datensatznummer angelegt. Das erste Byte des neuen Datensatzes enthält $FF (255), der Rest des Datensatzes wird mit $00 aufgefüllt. Steht der angesprochene Datensatz an Anfang eines Blocks, wird der Rest des Blocks ebenfalls mit leeren Datensätzen gefüllt. Jedesmal wenn ein nicht existierender Datensatz angesprochen wird, wird die Fehlermeldung '50, record not present' ausgegeben. Beim Schreiben eines neuen Datensatzes ist dies kein eigentlicher Fehler, sondern weist nur darauf hin, daß ein neuer Datensatz erzeugt wird. Diese Methode sollte man auch beim Anlegen einer neuen Datei benutzen, wenn man die maximale Zahl von Datensätzen kennt. Man setzt

einfach den Recordzeiger auf diesen Datensatz und schreibt $FF (CHR$(255)) in diesen Datensatz. Damit werden alle Datensätze bis zu dieser Nummer angelegt und die Fehlermeldung 50 tritt nicht mehr auf. Gleichzeitig weiß man auch, ob noch genügend Platz auf der Diskette ist. Ist dies nicht der Fall, erhält man die Fehlermeldung '52, file too large'.

Bei diesem Verfahren mit 6 Side-Sektoren kann eine relative Datei maximal 6 * 120 * 254 = 182 880 Bytes enthalten. Im Falle der VC 1541 ist dies mehr als die Kapazität der ganzen Diskette. Bei der größeren Floppy 8050, die pro Laufwerk mehr als 500 K abspeichern kann, bedeutet dies eine Einschränkung. Deshalb hat man ab der DOS-Version 2.7 eine Erweiterung des Side-Sektor-Verfahrens vorgenommen ('Super-Side-Sektor'), bei dem eine relative Datei maximal 23 MB an Daten enthalten kann. Dies ist bei der CBM 8250 und den Commodore-Festplatten sowie bei den neueren 8050-Floppies der Fall (siehe dazu auch Kapitel 5.2).

Da wie gesagt eine relative Datei zwei Datenkanäle erfordert, die VC 1541 jedoch nur drei Kanäle zur freien Verfügung hat, kann immer nur eine relative Datei offen sein. Der dritte Kanal könnte noch für eine gleichzeitig offene sequentielle Datei genutzt werden. Bei den großen CBM Floppies stehen Ihnen ebenfalls mehr Kanäle zur Verfügung (gleichzeitig 3 offene relative Dateien, siehe auch Kapitel 5.2).

```
******************************  LED am Laufwerk einschalten
C100   78         SEI
C101   A9 F7      LDA #$F7      LED-Bit löschen
C103   2D 00 1C   AND $1C00
C106   48         PHA
C107   A5 7F      LDA $7F       Drivenummer
C109   F0 05      BEQ $C110     0 ?
C10B   68         PLA
C10C   09 00      ORA #$00      nicht Laufwerk 0, dann LED aus
C10E   D0 03      BNE $C113
C110   68         PLA
C111   09 08      ORA #$08      LED einschalten
C113   8D 00 1C   STA $1C00
C116   58         CLI
C117   60         RTS

******************************  LED einschalten
C118   78         SEI
C119   A9 08      LDA #$08
C11B   0D 00 1C   ORA $1C00     LED ein
C11E   8D 00 1C   STA $1C00
C121   58         CLI
C122   60         RTS

******************************  Fehlerflags löschen
C123   A9 00      LDA #$00
C125   8D 6C 02   STA $026C
C128   8D 6D 02   STA $026D
C12B   60         RTS

******************************
C12C   78         SEI
C12D   8A         TXA           X-Register retten
C12E   48         PHA
C12F   A9 50      LDA #$50
C131   8D 6C 02   STA $026C
C134   A2 00      LDX #$00
C136   BD CA FE   LDA $FECA,X   8
C139   8D 6D 02   STA $026D
C13C   0D 00 1C   ORA $1C00
C13F   8D 00 1C   STA $1C00     LED einschalten
C142   68         PLA
C143   AA         TAX           X-Register zurückholen
C144   58         CLI
C145   60         RTS

******************************  Befehle vom Rechner auswerten
C146   A9 00      LDA #$00
C148   8D F9 02   STA $02F9
C14B   AD 8E 02   LDA $028E     letzte Drivenummer
C14E   85 7F      STA $7F       Drivenummer
C150   20 BC E6   JSR $E6BC     'ok'-Meldung bereitstellen
```

```
C153   A5 84      LDA $84       Sekundäradresse
C155   10 09      BPL $C160
C157   29 0F      AND #$0F
C159   C9 0F      CMP #$0F      15, Kommandokanal
C15B   F0 03      BEQ $C160     ja
C15D   4C B4 D7   JMP $D7B4     zum OPEN-Befehl
C160   20 B3 C2   JSR $C2B3     Zeilenlänge ermitteln und Flags löschen
C163   B1 A3      LDA ($A3),Y   erstes Zeichen holen
C165   8D 75 02   STA $0275     und merken
C168   A2 0B      LDX #$0B      11
C16A   BD 89 FE   LDA $FE89,X   Kommandos
C16D   CD 75 02   CMP $0275     mit erstem Zeichen vergleichen
C170   F0 08      BEQ $C17A     gefunden ?
C172   CA         DEX
C173   10 F5      BPL $C16A
C175   A9 31      LDA #$31      nicht gefunden
C177   4C C8 C1   JMP $C1C8     31, 'syntax error'
C17A   8E 2A 02   STX $022A     Nummer des Befehlsworts
C17D   E0 09      CPX #$09
C17F   90 03      BCC $C184     Befehlsnummer < 9 ?
C181   20 EE C1   JSR $C1EE     Test für 'R', 'S' und 'N'
C184   AE 2A 02   LDX $022A     Befehlsnummer
C187   BD 95 FE   LDA $FE95,X   Sprungadresse lo
C18A   85 6F      STA $6F
C18C   BD A1 FE   LDA $FEA1,X   Sprungadresse hi
C18F   85 70      STA $70
C191   6C 6F 00   JMP ($006F)   Sprung auf Befehl

******************************  Fehlermeldung nach Befehlsausführung bereitstellen
C194   A9 00      LDA #$00
C196   8D F9 02   STA $02F9
C199   AD 6C 02   LDA $026C     Flag gesetzt ?
C19C   D0 2A      BNE $C1C8     ja, dann Fehlermeldung setzen
C19E   A0 00      LDY #$00
C1A0   98         TYA           Fehlernummer 0
C1A1   84 80      STY $80       Tracknummer 0
C1A3   84 81      STY $81       Sektornummer 0
C1A5   84 A3      STY $A3
C1A7   20 C7 E6   JSR $E6C7     'ok'-Meldung bereitstellen
C1AA   20 23 C1   JSR $C123     Fehlerflags löschen
C1AD   A5 7F      LDA $7F       Drivenummer
C1AF   8D 8E 02   STA $028E     als letzte Drivenummer merken
C1B2   AA         TAX
C1B3   A9 00      LDA #$00
C1B5   95 FF      STA $FF,X
C1B7   20 BD C1   JSR $C1BD     Eingabepuffer löschen
C1BA   4C DA D4   JMP $D4DA     interne Kanäle schließen

******************************  Eingabepuffer löschen
C1BD   A0 28      LDY #$28      41 Zeichen löschen
C1BF   A9 00      LDA #$00
C1C1   99 00 02   STA $0200,Y   $200 bis $228
C1C4   88         DEY
C1C5   10 FA      BPL $C1C1
C1C7   60         RTS
```

```
******************************  Fehlermeldung ausgeben (Track + Sektor 0)
C1C8   A0 00      LDY #$00
C1CA   84 80      STY $80       Track = 0
C1CC   84 81      STY $81       Sektor = 0
C1CE   4C 45 E6   JMP $E645     Fehlernummer im Akku, Meldung generieren

******************************  Eingabezeile prüfen
C1D1   A2 00      LDX #$00
C1D3   8E 7A 02   STX $027A     Zeiger auf Laufwerknummer
C1D6   A9 3A      LDA #$3A      ':'
C1D8   20 68 C2   JSR $C268     Test der Zeile bis ':' oder bis zum Ende
C1DB   F0 05      BEQ $C1E2     kein Doppelpunkt gefunden ?
C1DD   88         DEY
C1DE   88         DEY
C1DF   8C 7A 02   STY $027A     zeigt auf Laufwerknummer (vor Doppelpunkt)
C1E2   4C 68 C3   JMP $C368     Laufwerknummer holen und LED einschalten

******************************  Eingabezeile prüfen
C1E5   A0 00      LDY #$00      Zeiger in Eingabepuffer
C1E7   A2 00      LDX #$00      Zähler für Kommas
C1E9   A9 3A      LDA #$3A      ':'
C1EB   4C 68 C2   JMP $C268     testet Zeile bis zum Doppelpunkt oder zum Ende

******************************  Eingabezeile prüfen
C1EE   20 E5 C1   JSR $C1E5     Zeilentest bis ':' oder Ende
C1F1   D0 05      BNE $C1F8     Doppelpunkt gefunden ?
C1F3   A9 34      LDA #$34
C1F5   4C C8 C1   JMP $C1C8     34, 'syntax error'
C1F8   88         DEY
C1F9   88         DEY           Zeiger vor den Doppelpunkt setzen
C1FA   8C 7A 02   STY $027A     als Position der Laufwerksnummer
C1FD   8A         TXA           Komma vor dem Doppelpunkt
C1FE   D0 F3      BNE $C1F3     ja, dann 'syntax error'
C200   A9 3D      LDA #$3D      '='
C202   20 68 C2   JSR $C268     Eingabe bis zum '=' prüfen
C205   8A         TXA           Komma gefunden ?
C206   F0 02      BEQ $C20A     nein
C208   A9 40      LDA #$40      Bit 6
C20A   09 21      ORA #$21      und Bit 0 und 5 setzen
C20C   8D 8B 02   STA $028B     Flag für Syntaxprüfung
C20F   E8         INX
C210   8E 77 02   STX $0277
C213   8E 78 02   STX $0278
C216   AD 8A 02   LDA $028A     Joker gefunden ?
C219   F0 0D      BEQ $C228     nein
C21B   A9 80      LDA #$80
C21D   0D 8B 02   ORA $028B     Bit 7 setzen
C220   8D 8B 02   STA $028B
C223   A9 00      LDA #$00
C225   8D 8A 02   STA $028A     und Jokerflag rücksetzen
C228   98         TYA           '=' gefunden ?
C229   F0 29      BEQ $C254     nein
C22B   9D 7A 02   STA $027A,X
C22E   AD 77 02   LDA $0277     Anzahl Kommas vor Gleichheitszeichen
C231   8D 79 02   STA $0279
C234   A9 8D      LDA #$8D      Shift CR
```

```
C236   20 68 C2   JSR $C268     Zeile bis zum Ende prüfen
C239   E8         INX           Kommazähler erhöhen
C23A   8E 78 02   STX $0278     Anzahl Kommas merken
C23D   CA         DEX
C23E   AD 8A 02   LDA $028A     Joker gefunden ?
C241   F0 02      BEQ $C245     nein
C243   A9 08      LDA #$08      Bit 3 setzen
C245   EC 77 02   CPX $0277     Komma nach dem Gleichheitszeichen ?
C248   F0 02      BEQ $C24C     nein
C24A   09 04      ORA #$04      Bit 2 setzen
C24C   09 03      ORA #$03      Bit 0 und 1 setzen
C24E   4D 8B 02   EOR $028B
C251   8D 8B 02   STA $028B     als Flag für Syntax-Prüfung
C254   AD 8B 02   LDA $028B     Syntaxflag
C257   AE 2A 02   LDX $022A     Befehlsnummer
C25A   3D A5 FE   AND $FEA5,X   mit Prüfbyte verknüpfen
C25D   D0 01      BNE $C260     fehlerhafte Syntax ?
C25F   60         RTS
C260   8D 6C 02   STA $026C     Fehlerflag setzen
C263   A9 30      LDA #$30
C265   4C C8 C1   JMP $C1C8     30, 'syntax error'

****************************   Zeichen im Eingabepuffer suchen
C268   8D 75 02   STA $0275     Zeichen merken
C26B   CC 74 02   CPY $0274     Zeile schon zu Ende ?
C26E   B0 2E      BCS $C29E     ja
C270   B1 A3      LDA ($A3),Y   Zeichen aus Puffer holen
C272   C8         INY
C273   CD 75 02   CMP $0275     mit gesuchtem Zeichen vergleichen
C276   F0 28      BEQ $C2A0     gefunden
C278   C9 2A      CMP #$2A      '*'
C27A   F0 04      BEQ $C280
C27C   C9 3F      CMP #$3F      '?'
C27E   D0 03      BNE $C283
C280   EE 8A 02   INC $028A     Jokerflag setzen
C283   C9 2C      CMP #$2C      ','
C285   D0 E4      BNE $C26B
C287   98         TYA
C288   9D 7B 02   STA $027B,X   Kommaposition merken
C28B   AD 8A 02   LDA $028A     Jokerflag
C28E   29 7F      AND #$7F
C290   F0 07      BEQ $C299     kein Joker
C292   A9 80      LDA #$80
C294   95 E7      STA $E7,X     Flag merken
C296   8D 8A 02   STA $028A     und als Jokerflag merken
C299   E8         INX           Kommazähler erhöhen
C29A   E0 04      CPX #$04      schon 4 Kommas ?
C29C   90 CD      BCC $C26B     nein, weitermachen
C29E   A0 00      LDY #$00
C2A0   AD 74 02   LDA $0274     Flag für Zeilenende setzen
C2A3   9D 7B 02   STA $027B,X
C2A6   AD 8A 02   LDA $028A     Jokerflag
C2A9   29 7F      AND #$7F
C2AB   F0 04      BEQ $C2B1     kein Joker
C2AD   A9 80      LDA #$80
C2AF   95 E7      STA $E7,X     Flag setzen
```

```
C2B1   98          TYA
C2B2   60          RTS

*****************************  Zeilenlänge prüfen
C2B3   A4 A3       LDY $A3     Zeiger in Befehlseingabepuffer
C2B5   F0 14       BEQ $C2CB   null ?
C2B7   88          DEY
C2B8   F0 10       BEQ $C2CA   eins ?
C2BA   B9 00 02    LDA $0200,Y Zeichen aus Eingabepuffer
C2BD   C9 0D       CMP #$0D    'CR'
C2BF   F0 0A       BEQ $C2CB   ja, Zeilenende
C2C1   88          DEY
C2C2   B9 00 02    LDA $0200,Y davorstehendes Zeichen
C2C5   C9 0D       CMP #$0D    'CR'
C2C7   F0 02       BEQ $C2CB   ja
C2C9   C8          INY
C2CA   C8          INY         Zeiger wieder auf alten Wert
C2CB   8C 74 02    STY $0274   gleich Zeilenlänge
C2CE   C0 2A       CPY #$2A    mit 42 Zeichen vergleichen
C2D0   A0 FF       LDY #$FF
C2D2   90 08       BCC $C2DC   kleiner, dann ok
C2D4   8C 2A 02    STY $022A
C2D7   A9 32       LDA #$32
C2D9   4C C8 C1    JMP $C1C8   32, 'syntax error' Zeile zu lang

*****************************  Flags für Befehlseingabe löschen
C2DC   A0 00       LDY #$00
C2DE   98          TYA
C2DF   85 A3       STA $A3     Zeiger auf Eingabepuffer Lo
C2E1   8D 58 02    STA $0258   Recordlänge
C2E4   8D 4A 02    STA $024A   Dateityp
C2E7   8D 96 02    STA $0296
C2EA   85 D3       STA $D3
C2EC   8D 79 02    STA $0279   Kommazähler
C2EF   8D 77 02    STA $0277       "
C2F2   8D 78 02    STA $0278       "
C2F5   8D 8A 02    STA $028A   Jokerflag
C2F8   8D 6C 02    STA $026C   Fehlerflag
C2FB   A2 05       LDX #$05
C2FD   9D 79 02    STA $0279,X Flags für Zeilenanalyse
C300   95 D7       STA $D7,X   Directory-Sektoren
C302   95 DC       STA $DC,X   Pufferzeiger
C304   95 E1       STA $E1,X   Drivenummern
C306   95 E6       STA $E6,X   Jokerflags
C308   9D 7F 02    STA $027F,X Tracknummern
C30B   9D 84 02    STA $0284,X Sektornummern
C30E   CA          DEX
C30F   D0 EC       BNE $C2FD
C311   60          RTS

*****************************  Drivenummer übernehmen
C312   AD 78 02    LDA $0278   Anzahl Kommas
C315   8D 77 02    STA $0277   merken
C318   A9 01       LDA #$01
C31A   8D 78 02    STA $0278   Anzahl der Drivenummern
C31D   8D 79 02    STA $0279
```

```
C320   AC 8E 02   LDY $028E      letzte Laufwerknummer
C323   A2 00      LDX #$00
C325   86 D3      STX $D3
C327   BD 7A 02   LDA $027A,X   Position des Doppelpunkts
C32A   20 3C C3   JSR $C33C     Drivenummer vor Doppelpunkt holen
C32D   A6 D3      LDX $D3
C32F   9D 7A 02   STA $027A,X   evtl. exakte Position abspeichern
C332   98         TYA
C333   95 E2      STA $E2,X     Drivenummer in Tabelle
C335   E8         INX
C336   EC 78 02   CPX $0278     schon alle Drivenummern geholt ?
C339   90 EA      BCC $C325     nein, weiter machen
C33B   60         RTS

******************************  Drivenummer suchen
C33C   AA         TAX           Position merken
C33D   A0 00      LDY #$00
C33F   A9 3A      LDA #$3A      ':'
C341   DD 01 02   CMP $0201,X   Doppelpunkt dahinter ?
C344   F0 0C      BEQ $C352     ja
C346   DD 00 02   CMP $0200,X   Doppelpunkt an dieser Stelle ?
C349   D0 16      BNE $C361     nein
C34B   E8         INX
C34C   98         TYA
C34D   29 01      AND #$01      Drivenummer
C34F   A8         TAY
C350   8A         TXA
C351   60         RTS

C352   BD 00 02   LDA $0200,X   Drivenummer holen
C355   E8         INX
C356   E8         INX
C357   C9 30      CMP #$30      '0' ?
C359   F0 F2      BEQ $C34D     ja
C35B   C9 31      CMP #$31      '1' ?
C35D   F0 EE      BEQ $C34D     ja
C35F   D0 EB      BNE $C34C     nein, letzte Drivenummer benutzen
C361   98         TYA           letzte Drivenummer
C362   09 80      ORA #$80      Bit 7 setzen, unsichere Drivenummer
C364   29 81      AND #$81      restliche Bits löschen
C366   D0 E7      BNE $C34F     Drivenummer zur Verfügung stellen

******************************  Drivenummer holen
C368   A9 00      LDA #$00
C36A   8D 8B 02   STA $028B     Syntaxflag löschen
C36D   AC 7A 02   LDY $027A     Position in Befehlszeile
C370   B1 A3      LDA ($A3),Y   Zeichen auf Befehslpuffer holen
C372   20 BD C3   JSR $C3BD     Laufwerknummer holen
C375   10 11      BPL $C388     sichere Nummer ?
C377   C8         INY           Zeiger erhöhen
C378   CC 74 02   CPY $0274     Zeilenende ?
C37B   B0 06      BCS $C383     ja
C37D   AC 74 02   LDY $0274
C380   88         DEY
C381   D0 ED      BNE $C370     Zeile nach Drivenummer absuchen
C383   CE 8B 02   DEC $028B
```

```
C386   A9 00      LDA #$00
C388   29 01      AND #$01
C38A   85 7F      STA $7F        Drivenummer
C38C   4C 00 C1   JMP $C100      LED einschalten

****************************  Drivenummer umschalten
C38F   A5 7F      LDA $7F        Drivenummer
C391   49 01      EOR #$01       Bit 0 umdrehen
C393   29 01      AND #$01
C395   85 7F      STA $7F
C397   60         RTS

****************************  Dateityp feststellen
C398   A0 00      LDY #$00
C39A   AD 77 02   LDA $0277      Gleichheitszeichen gefunden ?
C39D   CD 78 02   CMP $0278
C3A0   F0 16      BEQ $C3B8      nein
C3A2   CE 78 02   DEC $0278      Zeiger holen
C3A5   AC 78 02   LDY $0278
C3A8   B9 7A 02   LDA $027A,Y    Zeiger auf Zeichen hinter '=' setzen
C3AB   A8         TAY
C3AC   B1 A3      LDA ($A3),Y    Zeichen aus Puffer
C3AE   A0 04      LDY #$04       mit Kennzeichen für Filetyp vergleichen
C3B0   D9 BB FE   CMP $FEBB,Y    'S', 'P', 'U', 'R'
C3B3   F0 03      BEQ $C3B8      Übereinstimmung ?
C3B5   88         DEY
C3B6   D0 F8      BNE $C3B0
C3B8   98         TYA
C3B9   8D 96 02   STA $0296      Dateityp (1 bis 4) merken
C3BC   60         RTS

****************************  Drivenummer prüfen
C3BD   C9 30      CMP #$30       '0'
C3BF   F0 06      BEQ $C3C7
C3C1   C9 31      CMP #$31       '1'
C3C3   F0 02      BEQ $C3C7
C3C5   09 80      ORA #$80       keine null oder eins, dann Bit 7 setzen
C3C7   29 81      AND #$81
C3C9   60         RTS

****************************  Drivenummern überprüfen
C3CA   A9 00      LDA #$00
C3CC   85 6F      STA $6F
C3CE   8D 8D 02   STA $028D
C3D1   48         PHA
C3D2   AE 78 02   LDX $0278      Anzahl der Drivenummern
C3D5   68         PLA
C3D6   05 6F      ORA $6F
C3D8   48         PHA
C3D9   A9 01      LDA #$01
C3DB   85 6F      STA $6F
C3DD   CA         DEX
C3DE   30 0F      BMI $C3EF
C3E0   B5 E2      LDA $E2,X
C3E2   10 04      BPL $C3E8
C3E4   06 6F      ASL $6F
```

```
C3E6   06 6F      ASL $6F
C3E8   4A         LSR  A
C3E9   90 EA      BCC $C3D5
C3EB   06 6F      ASL $6F
C3ED   D0 E6      BNE $C3D5
C3EF   68         PLA
C3F0   AA         TAX
C3F1   BD 3F C4   LDA $C43F,X   Syntax-Flag holen
C3F4   48         PHA
C3F5   29 03      AND #$03
C3F7   8D 8C 02   STA $028C
C3FA   68         PLA
C3FB   0A         ASL  A
C3FC   10 3E      BPL $C43C
C3FE   A5 E2      LDA $E2
C400   29 01      AND #$01      Drivenummer isolieren
C402   85 7F      STA $7F
C404   AD 8C 02   LDA $028C
C407   F0 2B      BEQ $C434
C409   20 3D C6   JSR $C63D     Drive initialisieren
C40C   F0 12      BEQ $C420     kein Fehler ?
C40E   20 8F C3   JSR $C38F     auf anderes Drive umschalten
C411   A9 00      LDA #$00
C413   8D 8C 02   STA $028C
C416   20 3D C6   JSR $C63D     Drive initialisieren
C419   F0 1E      BEQ $C439     kein Fehler ?
C41B   A9 74      LDA #$74
C41D   20 C8 C1   JSR $C1C8     74, 'drive not ready'
C420   20 8F C3   JSR $C38F

C423   20 3D C6   JSR $C63D     Drive initialisieren
C426   08         PHP
C427   20 8F C3   JSR $C38F     auf anderes Drive umschalten
C42A   28         PLP
C42B   F0 0C      BEQ $C439     kein Fehler ?
C42D   A9 00      LDA #$00
C42F   8D 8C 02   STA $028C     Anzahl der Dives
C432   F0 05      BEQ $C439
C434   20 3D C6   JSR $C63D     Drive initialisieren
C437   D0 E2      BNE $C41B     Fehler ?
C439   4C 00 C1   JMP $C100     LED einschalten
C43C   2A         ROL  A        Drivenummer vom Carry nach Bit 0
C43D   4C 00 C4   JMP $C400

******************************  Flags für Drive-Prüfung
C440   00 80 41 01 01 01 01 81
C448   81 81 81 42 42 42 42

******************************  Datei im Directory suchen
C44F   20 CA C3   JSR $C3CA     Drive initialisieren
C452   A9 00      LDA #$00
C454   8D 92 02   STA $0292     Zeiger
C457   20 AC C5   JSR $C5AC     ersten Directoryblock lesen
C45A   D0 19      BNE $C475     Eintrag vorhanden ?
C45C   CE 8C 02   DEC $028C     Drivenummer klar ?
C45F   10 01      BPL $C462     nein
```

```
C461  60          RTS

C462  A9 01       LDA #$01
C464  8D 8D 02    STA $028D
C467  20 8F C3    JSR $C38F     Drive wechseln
C46A  20 00 C1    JSR $C100     LED einschalten
C46D  4C 52 C4    JMP $C452     und suchen

C470  20 17 C6    JSR $C617     nächste Datei im Directory suchen
C473  F0 10       BEQ $C485     nicht gefunden ?
C475  20 D8 C4    JSR $C4D8     Eintrag im Directory überprüfen
C478  AD 8F 02    LDA $028F
C47B  F0 01       BEQ $C47E     weitere Dateien ?
C47D  60          RTS

C47E  AD 53 02    LDA $0253
C481  30 ED       BMI $C470     Datei nicht gefunden ?
C483  10 F0       BPL $C475     ja
C485  AD 8F 02    LDA $028F
C488  F0 D2       BEQ $C45C
C48A  60          RTS

C48B  20 04 C6    JSR $C604     nächsten Directoryblock suchen
C48E  F0 1A       BEQ $C4AA     nicht gefunden ?
C490  D0 28       BNE $C4BA

C492  A9 01       LDA #$01
C494  8D 8D 02    STA $028D
C497  20 8F C3    JSR $C38F     Drive wechseln
C49A  20 00 C1    JSR $C100     LED einschalten
C49D  A9 00       LDA #$00
C49F  8D 92 02    STA $0292
C4A2  20 AC C5    JSR $C5AC     Directoryblock lesen
C4A5  D0 13       BNE $C4BA     gefunden ?
C4A7  8D 8F 02    STA $028F
C4AA  AD 8F 02    LDA $028F
C4AD  D0 28       BNE $C4D7
C4AF  CE 8C 02    DEC $028C
C4B2  10 DE       BPL $C492
C4B4  60          RTS

C4B5  20 17 C6    JSR $C617     nächster Eintrag im Directory
C4B8  F0 F0       BEQ $C4AA     nicht gefunden ?
C4BA  20 D8 C4    JSR $C4D8     Eintrag überprüfen
C4BD  AE 53 02    LDX $0253
C4C0  10 07       BPL $C4C9     Datei gefunden ?
C4C2  AD 8F 02    LDA $028F
C4C5  F0 EE       BEQ $C4B5     ja
C4C7  D0 0E       BNE $C4D7     nein, dann fertig

C4C9  AD 96 02    LDA $0296
C4CC  F0 09       BEQ $C4D7
C4CE  B5 E7       LDA $E7,X     Dateityp
C4D0  29 07       AND #$07
C4D2  CD 96 02    CMP $0296     gleich gesuchter Dateityp ?
C4D5  D0 DE       BNE $C4B5     nein
```

```
C4D7   60          RTS

C4D8   A2 FF       LDX #$FF
C4DA   8E 53 02    STX $0253      Flag für Datei gefunden
C4DD   E8          INX
C4DE   8E 8A 02    STX $028A
C4E1   20 89 C5    JSR $C589      Zeiger auf Datei setzen
C4E4   F0 06       BEQ $C4EC
C4E6   60          RTS

C4E7   20 94 C5    JSR $C594      Zeiger auf nächste Datei
C4EA   D0 FA       BNE $C4E6      Ende, dann fertig
C4EC   A5 7F       LDA $7F        Drivenummer
C4EE   55 E2       EOR $E2,X
C4F0   4A          LSR  A
C4F1   90 0B       BCC $C4FE
C4F3   29 40       AND #$40
C4F5   F0 F0       BEQ $C4E7
C4F7   A9 02       LDA #$02
C4F9   CD 8C 02    CMP $028C      Suche auf beiden Drives
C4FC   F0 E9       BEQ $C4E7      ja
C4FE   BD 7A 02    LDA $027A,X
C501   AA          TAX
C502   20 A6 C6    JSR $C6A6      Länge des Dateinamens holen
C505   A0 03       LDY #$03
C507   4C 1D C5    JMP $C51D

C50A   BD 00 02    LDA $0200,X    Zeichen aus Befehlszeile holen
C50D   D1 94       CMP ($94),Y    gleich Zeichen im Directory ?
C50F   F0 0A       BEQ $C51B      ja
C511   C9 3F       CMP #$3F       '?'
C513   D0 D2       BNE $C4E7      nein
C515   B1 94       LDA ($94),Y
C517   C9 A0       CMP #$A0       Shift Blank, Ende des Namens ?
C519   F0 CC       BEQ $C4E7      ja
C51B   E8          INX            Zeiger erhöhen
C51C   C8          INY
C51D   EC 76 02    CPX $0276      Ende des Namens im Befehl ?
C520   B0 09       BCS $C52B      ja
C522   BD 00 02    LDA $0200,X    nächstes Zeichen
C525   C9 2A       CMP #$2A       '*'
C527   F0 0C       BEQ $C535      ja, Datei gefunden
C529   D0 DF       BNE $C50A      sonst weitersuchen

C52B   C0 13       CPY #$13       19
C52D   B0 06       BCS $C535      Ende des Namens erreicht
C52F   B1 94       LDA ($94),Y
C531   C9 A0       CMP #$A0       Shift Blank, Ende des Namens
C533   D0 B2       BNE $C4E7      nicht gefunden
C535   AE 79 02    LDX $0279
C538   8E 53 02    STX $0253
C53B   B5 E7       LDA $E7,X
C53D   29 80       AND #$80
C53F   8D 8A 02    STA $028A
C542   AD 94 02    LDA $0294
C545   95 DD       STA $DD,X
```

```
C547   A5 81      LDA $81        Sektornummer des Directorys
C549   95 D8      STA $D8,X      in Tabelle eintragen
C54B   A0 00      LDY #$00
C54D   B1 94      LDA ($94),Y    Filetyp
C54F   C8         INY
C550   48         PHA
C551   29 40      AND #$40       Scratchschutzbit (6) isolieren
C553   85 6F      STA $6F        und merken
C555   68         PLA
C556   29 DF      AND #$DF       Bit 7 löschen
C558   30 02      BMI $C55C
C55A   09 20      ORA #$20       Bit 5 setzen
C55C   29 27      AND #$27       Bit 3 und 4 löschen
C55E   05 6F      ORA $6F        Bit 6 wiederholen
C560   85 6F      STA $6F
C562   A9 80      LDA #$80
C564   35 E7      AND $E7,X      Flag für Joker isolieren
C566   05 6F      ORA $6F
C568   95 E7      STA $E7,X      in Tabelle schreiben
C56A   B5 E2      LDA $E2,X
C56C   29 80      AND #$80
C56E   05 7F      ORA $7F        Drivenummer
C570   95 E2      STA $E2,X
C572   B1 94      LDA ($94),Y
C574   9D 80 02   STA $0280,X    erstes Track der Datei
C577   C8         INY
C578   B1 94      LDA ($94),Y
C57A   9D 85 02   STA $0285,X    und Sektor aus Directory holen
C57D   AD 58 02   LDA $0258      Recordlänge
C580   D0 07      BNE $C589      schon erfaßt ?
C582   A0 15      LDY #$15
C584   B1 94      LDA ($94),Y    Recordlänge
C586   8D 58 02   STA $0258      aus Directory holen
C589   A9 FF      LDA #$FF
C58B   8D 8F 02   STA $028F
C58E   AD 78 02   LDA $0278
C591   8D 79 02   STA $0279
C594   CE 79 02   DEC $0279
C597   10 01      BPL $C59A
C599   60         RTS

C59A   AE 79 02   LDX $0279
C59D   B5 E7      LDA $E7,X      Flag für Joker gesetzt ?
C59F   30 05      BMI $C5A6      ja
C5A1   BD 80 02   LDA $0280,X    Tracknummer schon gesetzt ?
C5A4   D0 EE      BNE $C594      ja
C5A6   A9 00      LDA #$00
C5A8   8D 8F 02   STA $028F
C5AB   60         RTS

C5AC   A0 00      LDY #$00
C5AE   8C 91 02   STY $0291
C5B1   88         DEY
C5B2   8C 53 02   STY $0253
C5B5   AD 85 FE   LDA $FE85      18, Directorytrack
C5B8   85 80      STA $80
```

```
C5BA   A9 01      LDA #$01
C5BC   85 81      STA $81       Sektor 1
C5BE   8D 93 02   STA $0293
C5C1   20 75 D4   JSR $D475     Sektor lesen
C5C4   AD 93 02   LDA $0293
C5C7   D0 01      BNE $C5CA
C5C9   60         RTS

C5CA   A9 07      LDA #$07
C5CC   8D 95 02   STA $0295     Anzahl der Directoryeinträge (-1)
C5CF   A9 00      LDA #$00
C5D1   20 F6 D4   JSR $D4F6     Zeichen aus Puffer holen
C5D4   8D 93 02   STA $0293     als Tracknummer merken
C5D7   20 E8 D4   JSR $D4E8     Pufferzeiger setzen
C5DA   CE 95 02   DEC $0295     Zähler vermindern
C5DD   A0 00      LDY #$00
C5DF   B1 94      LDA ($94),Y   erstes Byte aus Directory
C5E1   D0 18      BNE $C5FB
C5E3   AD 91 02   LDA $0291
C5E6   D0 2F      BNE $C617
C5E8   20 3B DE   JSR $DE3B     Track und Sektornummer holen
C5EB   A5 81      LDA $81
C5ED   8D 91 02   STA $0291     Sektornummer
C5F0   A5 94      LDA $94
C5F2   AE 92 02   LDX $0292
C5F5   8D 92 02   STA $0292     Pufferzeiger
C5F8   F0 1D      BEQ $C617
C5FA   60         RTS

C5FB   A2 01      LDX #$01
C5FD   EC 92 02   CPX $0292     Pufferzeiger auf eins ?
C600   D0 2D      BNE $C62F
C602   F0 13      BEQ $C617

C604   AD 85 FE   LDA $FE85     18, Tracknummer der BAM
C607   85 80      STA $80       Tracknummer
C609   AD 90 02   LDA $0290
C60C   85 81      STA $81       Sektornummer
C60E   20 75 D4   JSR $D475     Block lesen
C611   AD 94 02   LDA $0294
C614   20 C8 D4   JSR $D4C8     Pufferzeiger setzen
C617   A9 FF      LDA #$FF
C619   8D 53 02   STA $0253     Flag für Datei gefunden löschen
C61C   AD 95 02   LDA $0295
C61F   30 08      BMI $C629     schon alle Directoryeinträge geprüft ?
C621   A9 20      LDA #$20
C623   20 C6 D1   JSR $D1C6     Pufferzeiger um 32 erhöhen, nächster Eintrag
C626   4C D7 C5   JMP $C5D7     und weitersuchen

C629   20 4D D4   JSR $D44D     Pufferzeiger setzen
C62C   4C C4 C5   JMP $C5C4     nächsten Block lesen

C62F   A5 94      LDA $94
C631   8D 94 02   STA $0294
C634   20 3B DE   JSR $DE3B     Track und Sektornummer aus Puffer holen
C637   A5 81      LDA $81
```

```
C639   8D 90 02   STA $0290      Sektornummer merken
C63C   60          RTS

*****************************  Drive testen und initialisieren
C63D   A5 68       LDA $68
C63F   D0 28       BNE $C669
C641   A6 7F       LDX $7F       Drivenummer
C643   56 1C       LSR $1C,X     wurde Diskette gewechselt ?
C645   90 22       BCC $C669     nein, dann fertig
C647   A9 FF       LDA #$FF
C649   8D 98 02    STA $0298     Fehlerflag setzen
C64C   20 0E D0    JSR $D00E     Directorytrack lesen
C64F   A0 FF       LDY #$FF
C651   C9 02       CMP #$02      20, 'read error' ?
C653   F0 0A       BEQ $C65F     ja
C655   C9 03       CMP #$03      21, 'read error' ?
C657   F0 06       BEQ $C65F     ja
C659   C9 0F       CMP #$0F      74, 'drive not ready' ?
C65B   F0 02       BEQ $C65F     ja
C65D   A0 00       LDY #$00
C65F   A6 7F       LDX $7F       Drivenummer
C661   98          TYA
C662   95 FF       STA $FF,X     Fehlerflag merken
C664   D0 03       BNE $C669     Fehler ?
C666   20 42 D0    JSR $D042     BAM laden
C669   A6 7F       LDX $7F       Drivenummer
C66B   B5 FF       LDA $FF,X     Fehlerkode übergeben
C66D   60          RTS

*****************************  Name der Datei in Directorypuffer
C66E   48          PHA
C66F   20 A6 C6    JSR $C6A6     Ende des Namens holen
C672   20 88 C6    JSR $C688     Filenamen in Puffer schreiben
C675   68          PLA
C676   38          SEC
C677   ED 4B 02    SBC $024B     Länge mit maximaler Länge vergleichen
C67A   AA          TAX
C67B   F0 0A       BEQ $C687
C67D   90 08       BCC $C687
C67F   A9 A0       LDA #$A0      mit 'Shift Blank' auffüllen
C681   91 94       STA ($94),Y
C683   C8          INY
C684   CA          DEX
C685   D0 FA       BNE $C681
C687   60          RTS

*****************************
C688   98          TYA           Puffernummer
C689   0A          ASL  A
C68A   A8          TAY           mal 2 als Zeiger
C68B   B9 99 00    LDA $0099,Y
C68E   85 94       STA $94
C690   B9 9A 00    LDA $009A,Y   Pufferzeiger nach $94/$95
C693   85 95       STA $95
C695   A0 00       LDY #$00
C697   BD 00 02    LDA $0200,X   Zeichen in Puffer übertragen
```

```
C69A   91 94      STA ($94),Y
C69C   C8         INY
C69D   F0 06      BEQ $C6A5     Puffer bereits voll ?
C69F   E8         INX
C6A0   EC 76 02   CPX $0276
C6A3   90 F2      BCC $C697
C6A5   60         RTS

****************************  Ende des Namens im Befehl suchen
C6A6   A9 00      LDA #$00
C6A8   8D 4B 02   STA $024B     Vorbesetzung für Länge
C6AB   8A         TXA
C6AC   48         PHA
C6AD   BD 00 02   LDA $0200,X   Zeichen aus Puffer holen
C6B0   C9 2C      CMP #$2C      ','
C6B2   F0 14      BEQ $C6C8
C6B4   C9 3D      CMP #$3D      '='
C6B6   F0 10      BEQ $C6C8
C6B8   EE 4B 02   INC $024B     Länge des Namens erhöhen
C6BB   E8         INX
C6BC   A9 0F      LDA #$0F      15
C6BE   CD 4B 02   CMP $024B
C6C1   90 05      BCC $C6C8     größer ?
C6C3   EC 74 02   CPX $0274     Eingabezeile zu Ende ?
C6C6   90 E5      BCC $C6AD
C6C8   8E 76 02   STX $0276
C6CB   68         PLA
C6CC   AA         TAX           Zeiger auf Ende des Namens
C6CD   60         RTS

****************************
C6CE   A5 83      LDA $83
C6D0   48         PHA           Sekundäradresse und Kanalnummer
C6D1   A5 82      LDA $82
C6D3   48         PHA
C6D4   20 DE C6   JSR $C6DE     Dateieintrag für Directory erzeugen
C6D7   68         PLA
C6D8   85 82      STA $82
C6DA   68         PLA           Daten zurückholen
C6DB   85 83      STA $83
C6DD   60         RTS

****************************
C6DE   A9 11      LDA #$11      17
C6E0   85 83      STA $83       Sekundäradresse
C6E2   20 EB D0   JSR $D0EB     Kanal zum Lesen öffnen
C6E5   20 E8 D4   JSR $D4E8     Pufferzeiger setzen
C6E8   AD 53 02   LDA $0253
C6EB   10 0A      BPL $C6F7     noch nicht letzter Eintrag ?
C6ED   AD 8D 02   LDA $028D
C6F0   D0 0A      BNE $C6FC
C6F2   20 06 C8   JSR $C806     'blocks free.' schreiben
C6F5   18         CLC
C6F6   60         RTS
C6F7   AD 8D 02   LDA $028D
C6FA   F0 1F      BEQ $C71B
```

```
C6FC  CE 8D 02   DEC $028D
C6FF  D0 0D      BNE $C70E
C701  CE 8D 02   DEC $028D
C704  20 8F C3   JSR $C38F     Drive wechseln
C707  20 06 C8   JSR $C806     'blocks free.' schreiben
C70A  38         SEC
C70B  4C 8F C3   JMP $C38F     Drive wechseln

C70E  A9 00      LDA #$00
C710  8D 73 02   STA $0273     Drivenummer für Überschrift, Hi-Byte
C713  8D 8D 02   STA $028D
C716  20 B7 C7   JSR $C7B7     Überschrift schreiben
C719  38         SEC
C71A  60         RTS

C71B  A2 18      LDX #$18
C71D  A0 1D      LDY #$1D
C71F  B1 94      LDA ($94),Y   Anzahl Blocks hi
C721  8D 73 02   STA $0273     in Puffer
C724  F0 02      BEQ $C728     null ?
C726  A2 16      LDX #$16
C728  88         DEY
C729  B1 94      LDA ($94),Y   Anzahl Blocks lo
C72B  8D 72 02   STA $0272     in Puffer
C72E  E0 16      CPX #$16
C730  F0 0A      BEQ $C73C
C732  C9 0A      CMP #$0A      10
C734  90 06      BCC $C73C
C736  CA         DEX
C737  C9 64      CMP #$64      100
C739  90 01      BCC $C73C
C73B  CA         DEX
C73C  20 AC C7   JSR $C7AC     Puffer löschen
C73F  B1 94      LDA ($94),Y   Filetyp
C741  48         PHA
C742  0A         ASL  A        Bit 7 ins Carry
C743  10 05      BPL $C74A     Bit 6 nicht gesetzt ?
C745  A9 3C      LDA #$3C      '<' für geschütztes File
C747  9D B2 02   STA $02B2,X   hinter Filetyp schreiben
C74A  68         PLA
C74B  29 0F      AND #$0F      Bit 0 bis 3 isolieren
C74D  A8         TAY           als Index als Filetypbezeichnungen
C74E  B9 C5 FE   LDA $FEC5,Y   3. Buchstabe des Filetyps
C751  9D B1 02   STA $02B1,X   in Puffer
C754  CA         DEX
C755  B9 C0 FE   LDA $FEC0,Y   2. Buchstabe des Filetyps
C758  9D B1 02   STA $02B1,X   in Puffer
C75B  CA         DEX
C75C  B9 BB FE   LDA $FEBB,Y   1. Buchstabe des Filetyps
C75F  9D B1 02   STA $02B1,X   in Puffer
C762  CA         DEX
C763  CA         DEX
C764  B0 05      BCS $C76B     File nicht geschlossen ?
C766  A9 2A      LDA #$2A      '*'
C768  9D B2 02   STA $02B2,X   vor Filetyp in Puffer
C76B  A9 A0      LDA #$A0      mit 'shift blank' auffüllen
```

```
C76D   9D B1 02   STA $02B1,X   in Puffer
C770   CA         DEX
C771   A0 12      LDY #$12
C773   B1 94      LDA ($94),Y   Filenamen
C775   9D B1 02   STA $02B1,X   in Puffer schreiben
C778   CA         DEX
C779   88         DEY
C77A   C0 03      CPY #$03
C77C   B0 F5      BCS $C773
C77E   A9 22      LDA #$22      '"'
C780   9D B1 02   STA $02B1,X   vor Filenamen schreiben
C783   E8         INX
C784   E0 20      CPX #$20
C786   B0 0B      BCS $C793
C788   BD B1 02   LDA $02B1,X   Zeichen aus Puffer
C78B   C9 22      CMP #$22      '"' ?
C78D   F0 04      BEQ $C793
C78F   C9 A0      CMP #$A0      'Shift Blank' am Ende des Namens
C791   D0 F0      BNE $C783
C793   A9 22      LDA #$22      durch '"'  ersetzen
C795   9D B1 02   STA $02B1,X
C798   E8         INX
C799   E0 20      CPX #$20
C79B   B0 0A      BCS $C7A7
C79D   A9 7F      LDA #$7F      Bit 7
C79F   3D B1 02   AND $02B1,X
C7A2   9D B1 02   STA $02B1,X   in den restlichen Zeichen löschen
C7A5   10 F1      BPL $C798
C7A7   20 B5 C4   JSR $C4B5     nächsten Directoryeintrag suchen
C7AA   38         SEC
C7AB   60         RTS

******************************  Puffer für Directory löschen
C7AC   A0 1B      LDY #$1B
C7AE   A9 20      LDA #$20      ' ' Blank
C7B0   99 B0 02   STA $02B0,Y   in Puffer schreiben
C7B3   88         DEY
C7B4   D0 FA      BNE $C7B0
C7B6   60         RTS

******************************  Überschrift mit Diskettenname erzeugen
C7B7   20 19 F1   JSR $F119     bei Bedarf initialisieren
C7BA   20 DF F0   JSR $F0DF     Diskname lesen
C7BD   20 AC C7   JSR $C7AC     Puffer löschen
C7C0   A9 FF      LDA #$FF
C7C2   85 6F      STA $6F
C7C4   A6 7F      LDX $7F       Drivenummer
C7C6   8E 72 02   STX $0272     als Blockzahl lo in Puffer
C7C9   A9 00      LDA #$00
C7CB   8D 73 02   STA $0273     Blockzahl lo
C7CE   A6 F9      LDX $F9       Puffernummer
C7D0   BD E0 FE   LDA $FEE0,X   Hi-Byte der Pufferadresse
C7D3   85 95      STA $95
C7D5   AD 88 FE   LDA $FE88     $90, Position des Diskettennames
C7D8   85 94      STA $94       merken
C7DA   A0 16      LDY #$16
```

```
C7DC   B1 94      LDA ($94),Y   Puffer mit 'Shift blank' füllen
C7DE   C9 A0      CMP #$A0
C7E0   D0 0B      BNE $C7ED
C7E2   A9 31      LDA #$31      '1'
C7E4   2C         .BYTE $2C
C7E5   B1 94      LDA ($94),Y   Zeichen aus Puffer
C7E7   C9 A0      CMP #$A0      mit 'Shift blank' vergleichen
C7E9   D0 02      BNE $C7ED
C7EB   A9 20      LDA #$20      ' ' Blank
C7ED   99 B3 02   STA $02B3,Y   in Puffer
C7F0   88         DEY
C7F1   10 F2      BPL $C7E5
C7F3   A9 12      LDA #$12      'RVS ON'
C7F5   8D B1 02   STA $02B1     in Puffer
C7F8   A9 22      LDA #$22      '"'
C7FA   8D B2 02   STA $02B2     vor und
C7FD   8D C3 02   STA $02C3     hinter Diskname schreiben
C800   A9 20      LDA #$20      ' ' Blank
C802   8D C4 02   STA $02C4     dahinter
C805   60         RTS

******************************  Svhlußzeile erzeugen
C806   20 AC C7   JSR $C7AC     Puffer löschen
C809   A0 0B      LDY #$0B      12 Zeichen
C80B   B9 17 C8   LDA $C817,Y   'blocks free.'
C80E   99 B1 02   STA $02B1,Y   in Puffer schreiben
C811   88         DEY
C812   10 F7      BPL $C80B
C814   4C 4D EF   JMP $EF4D     Zahl der freien Blöcke davor

******************************
C817   42 4C 4F 43 4B 53 20 46  'blocks f'
C81F   52 45 45 2E              'ree.'

******************************  S-Befehl   'Scratch'
C823   20 98 C3   JSR $C398     Dateityp ermitteln
C826   20 20 C3   JSR $C320     Drivenummer holen
C829   20 CA C3   JSR $C3CA     Drive bei Bedarf initialisieren
C82C   A9 00      LDA #$00
C82E   85 86      STA $86       Zähler für gelöschte Dateien
C830   20 9D C4   JSR $C49D     Datei im Directory suchen
C833   30 3D      BMI $C872     nicht gefunden ?
C835   20 B7 DD   JSR $DDB7     ist Datei offen ?
C838   90 33      BCC $C86D     ja
C83A   A0 00      LDY #$00
C83C   B1 94      LDA ($94),Y   Filetyp
C83E   29 40      AND #$40      Scratch-Schutz ?
C840   D0 2B      BNE $C86D     ja
C842   20 B6 C8   JSR $C8B6     Datei löschen und in Directory vermerken
C845   A0 13      LDY #$13
C847   B1 94      LDA ($94),Y   Tracknummer des ersten Side-Sektors
C849   F0 0A      BEQ $C855     keiner vorhanden ?
C84B   85 80      STA $80       Tracknummer merken
C84D   C8         INY
C84E   B1 94      LDA ($94),Y   und Sektornummer
C850   85 81      STA $81
```

```
C852   20 7D C8   JSR $C87D     Side-Sektoren löschen
C855   AE 53 02   LDX $0253     Dateinummer
C858   A9 20      LDA #$20
C85A   35 E7      AND $E7,X     Bit 5 gesetzt ?
C85C   D0 0D      BNE $C86B     ja, Datei nicht geschlossen
C85E   BD 80 02   LDA $0280,X   Track
C861   85 80      STA $80
C863   BD 85 02   LDA $0285,X   und Sektor holen
C866   85 81      STA $81
C868   20 7D C8   JSR $C87D     Datei löschen
C86B   E6 86      INC $86       Anzahl der gelöschten Dateien erhöhen
C86D   20 8B C4   JSR $C48B     nächste Datei suchen
C870   10 C3      BPL $C835     falls vorhanden löschen
C872   A5 86      LDA $86       Anzahl der gelöschten Files
C874   85 80      STA $80       als 'Track' speichern
C876   A9 01      LDA #$01      1 als Disk-Status
C878   A0 00      LDY #$00      0 als 'Sektor'
C87A   4C A3 C1   JMP $C1A3     Meldung 'files scratched' bereit stellen

******************************  Datei löschen
C87D   20 5F EF   JSR $EF5F     Block in BAM freigeben
C880   20 75 D4   JSR $D475
C883   20 19 F1   JSR $F119     Puffernummer der BAM holen
C886   B5 A7      LDA $A7,X
C888   C9 FF      CMP #$FF
C88A   F0 08      BEQ $C894
C88C   AD F9 02   LDA $02F9
C88F   09 40      ORA #$40
C891   8D F9 02   STA $02F9
C894   A9 00      LDA #$00
C896   20 C8 D4   JSR $D4C8     Pufferzeiger auf Null
C899   20 56 D1   JSR $D156     Track holen
C89C   85 80      STA $80
C89E   20 56 D1   JSR $D156     Sektor holen
C8A1   85 81      STA $81
C8A3   A5 80      LDA $80       Tracknummer
C8A5   D0 06      BNE $C8AD     ungleich Null ?
C8A7   20 F4 EE   JSR $EEF4     BAM schreiben
C8AA   4C 27 D2   JMP $D227     Kanal schließen

C8AD   20 5F EF   JSR $EF5F     Block in BAM freigeben
C8B0   20 4D D4   JSR $D44D     nächsten Block lesen
C8B3   4C 94 C8   JMP $C894     und weiter machen

******************************  Directoryeintrag löschen
C8B6   A0 00      LDY #$00
C8B8   98         TYA
C8B9   91 94      STA ($94),Y   Filetyp auf Null setzen
C8BB   20 5E DE   JSR $DE5E     Block schreiben
C8BE   4C 99 D5   JMP $D599     und prüfen

******************************  D-Befehl,   'Backup'
C8C1   A9 31      LDA #$31
C8C3   4C C8 C1   JMP $C1C8     31, 'syntax error'

******************************  Diskette formatieren
```

```
C8C6   A9 4C      LDA #$4C      JMP-Befehl
C8C8   8D 00 06   STA $0600
C8CB   A9 C7      LDA #$C7
C8CD   8D 01 06   STA $0601     JMP $FAC7 nach $600 bis $602
C8D0   A9 FA      LDA #$FA
C8D2   8D 02 06   STA $0602
C8D5   A9 03      LDA #$03
C8D7   20 D3 D6   JSR $D6D3     Track und Sektornummer setzen
C8DA   A5 7F      LDA $7F       Drivenummer
C8DC   09 E0      ORA #$E0      Befehlskode für Formatieren
C8DE   85 03      STA $03       übergeben
C8E0   A5 03      LDA $03
C8E2   30 FC      BMI $C8E0     warten auf Ende der Formatierung
C8E4   C9 02      CMP #$02      Rückmeldung prüfen
C8E6   90 07      BCC $C8EF     kleiner 2, dann ok
C8E8   A9 03      LDA #$03
C8EA   A2 00      LDX #$00
C8EC   4C 0A E6   JMP $E60A     21, 'read error'
C8EF   60         RTS

****************************** C-Befehl,   'Copy'
C8F0   A9 E0      LDA #$E0
C8F2   8D 4F 02   STA $024F
C8F5   20 D1 F0   JSR $F0D1
C8F8   20 19 F1   JSR $F119     Puffernummer der BAM holen
C8FB   A9 FF      LDA #$FF
C8FD   95 A7      STA $A7,X
C8FF   A9 0F      LDA #$0F
C901   8D 56 02   STA $0256
C904   20 E5 C1   JSR $C1E5     Eingabezeile prüfen
C907   D0 03      BNE $C90C
C909   4C C1 C8   JMP $C8C1     31, 'syntax error'

C90C   20 F8 C1   JSR $C1F8     Eingabezeile prüfen
C90F   20 20 C3   JSR $C320     Drivenummern testen
C912   AD 8B 02   LDA $028B     Flag für Syntaxprüfung
C915   29 55      AND #$55
C917   D0 0F      BNE $C928
C919   AE 7A 02   LDX $027A
C91C   BD 00 02   LDA $0200,X   Zeichen des Befehls
C91F   C9 2A      CMP #$2A      '*'
C921   D0 05      BNE $C928
C923   A9 30      LDA #$30
C925   4C C8 C1   JMP $C1C8     30, 'syntax error'

C928   AD 8B 02   LDA $028B     Syntaxflag
C92B   29 D9      AND #$D9
C92D   D0 F4      BNE $C923     30,'syntax error'
C92F   4C 52 C9   JMP $C952

C932   A9 00      LDA #$00
C934   8D 58 02   STA $0258
C937   8D 8C 02   STA $028C     Anzahl der Laufwerke
C93A   8D 80 02   STA $0280     Tracknummer im Directory
C93D   8D 81 02   STA $0281
C940   A5 E3      LDA $E3
```

```
C942   29 01      AND #$01
C944   85 7F      STA $7F       Drivenummer
C946   09 01      ORA #$01
C948   8D 91 02   STA $0291
C94B   AD 7B 02   LDA $027B
C94E   8D 7A 02   STA $027A
C951   60         RTS

C952   20 4F C4   JSR $C44F     Datei im Directory suchen
C955   AD 78 02   LDA $0278     Anzahl der Dateinamen im Befehl
C958   C9 03      CMP #$03      kleiner als drei ?
C95A   90 45      BCC $C9A1     ja
C95C   A5 E2      LDA $E2       erste Drivenummer
C95E   C5 E3      CMP $E3       zweite Drivenummer
C960   D0 3F      BNE $C9A1     nicht auf gleichem Laufwerk ?
C962   A5 DD      LDA $DD       Directoryblock der ersten Datei
C964   C5 DE      CMP $DE       gleich Directoryblock der zweiten Datei ?
C966   D0 39      BNE $C9A1     nein
C968   A5 D8      LDA $D8       Directorysektor der ersten Datei
C96A   C5 D9      CMP $D9       gleich Directorysektor der zweiten Datei ?
C96C   D0 33      BNE $C9A1     nein
C96E   20 CC CA   JSR $CACC     ist Datei vorhanden ?
C971   A9 01      LDA #$01
C973   8D 79 02   STA $0279
C976   20 FA C9   JSR $C9FA
C979   20 25 D1   JSR $D125     Dateityp holen
C97C   F0 04      BEQ $C982     Rel-Datei ?
C97E   C9 02      CMP #$02      Prg-Datei
C980   D0 05      BNE $C987     nein
C982   A9 64      LDA #$64
C984   20 C8 C1   JSR $C1C8     64, 'file type mismatch'
C987   A9 12      LDA #$12      18
C989   85 83      STA $83       Sekundäradresse
C98B   AD 3C 02   LDA $023C
C98E   8D 3D 02   STA $023D
C991   A9 FF      LDA #$FF
C993   8D 3C 02   STA $023C
C996   20 2A DA   JSR $DA2A     Append vorbereiten
C999   A2 02      LDX #$02
C99B   20 B9 C9   JSR $C9B9     Dateien kopieren
C99E   4C 94 C1   JMP $C194     fertig

C9A1   20 A7 C9   JSR $C9A7     Dateien kopieren
C9A4   4C 94 C1   JMP $C194     fertig

C9A7   20 E7 CA   JSR $CAE7
C9AA   A5 E2      LDA $E2       Drivenummer des ersten Files
C9AC   29 01      AND #$01
C9AE   85 7F      STA $7F       Drivenummer
C9B0   20 86 D4   JSR $D486     Block anlegen
C9B3   20 E4 D6   JSR $D6E4     Datei in Directory eintragen
C9B6   AE 77 02   LDX $0277
C9B9   8E 79 02   STX $0279
C9BC   20 FA C9   JSR $C9FA
C9BF   A9 11      LDA #$11      17
C9C1   85 83      STA $83
```

```
C9C3   20 EB D0   JSR $D0EB
C9C6   20 25 D1   JSR $D125     Dateityp holen
C9C9   D0 03      BNE $C9CE     keine Rel-Datei ?
C9CB   20 53 CA   JSR $CA53
C9CE   A9 08      LDA #$08
C9D0   85 F8      STA $F8
C9D2   4C D8 C9   JMP $C9D8

C9D5   20 9B CF   JSR $CF9B     Byte in Puffer schreiben
C9D8   20 35 CA   JSR $CA35     und Byte holen
C9DB   A9 80      LDA #$80
C9DD   20 A6 DD   JSR $DDA6     Bit 7 testen
C9E0   F0 F3      BEQ $C9D5     nicht gesetzt ?
C9E2   20 25 D1   JSR $D125     Dateityp prüfen
C9E5   F0 03      BEQ $C9EA     Rel-Datei ?
C9E7   20 9B CF   JSR $CF9B     Datenbyte in Puffer holen
C9EA   AE 79 02   LDX $0279
C9ED   E8         INX
C9EE   EC 78 02   CPX $0278
C9F1   90 C6      BCC $C9B9
C9F3   A9 12      LDA #$12      18
C9F5   85 83      STA $83
C9F7   4C 02 DB   JMP $DB02     Kanal schließen

C9FA   AE 79 02   LDX $0279
C9FD   B5 E2      LDA $E2,X     Drivenummer
C9FF   29 01      AND #$01
CA01   85 7F      STA $7F       merken
CA03   AD 85 FE   LDA $FE85     18, Directorytrack
CA06   85 80      STA $80       merken
CA08   B5 D8      LDA $D8,X     Directorysektor
CA0A   85 81      STA $81
CA0C   20 75 D4   JSR $D475     Block lesen
CA0F   AE 79 02   LDX $0279
CA12   B5 DD      LDA $DD,X     Zeiger in Block
CA14   20 C8 D4   JSR $D4C8     Pufferzeiger setzen
CA17   AE 79 02   LDX $0279
CA1A   B5 E7      LDA $E7,X     Dateityp
CA1C   29 07      AND #$07      isolieren
CA1E   8D 4A 02   STA $024A     und merken
CA21   A9 00      LDA #$00
CA23   8D 58 02   STA $0258
CA26   20 A0 D9   JSR $D9A0     Parameter für Rel-Datei holen
CA29   A0 01      LDY #$01
CA2B   20 25 D1   JSR $D125     Dateityp holen
CA2E   F0 01      BEQ $CA31     Rel-Datei ?
CA30   C8         INY
CA31   98         TYA
CA32   4C C8 D4   JMP $D4C8     Pufferzeiger setzen

CA35   A9 11      LDA #$11      17
CA37   85 83      STA $83
CA39   20 9B D3   JSR $D39B     Kanal öffnen und Byte holen
CA3C   85 85      STA $85
CA3E   A6 82      LDX $82       Kanalnummer
CA40   B5 F2      LDA $F2,X
```

```
CA42   29 08      AND #$08      Endekennzeichen isolieren
CA44   85 F8      STA $F8
CA46   D0 0A      BNE $CA52     nicht gesetzt ?
CA48   20 25 D1   JSR $D125     Dateityp holen
CA4B   F0 05      BEQ $CA52     Rel-Datei ?
CA4D   A9 80      LDA #$80
CA4F   20 97 DD   JSR $DD97     Bit 7 setzen
CA52   60         RTS

CA53   20 D3 D1   JSR $D1D3     Drivenummer setzen
CA56   20 CB E1   JSR $E1CB
CA59   A5 D6      LDA $D6
CA5B   48         PHA
CA5C   A5 D5      LDA $D5
CA5E   48         PHA
CA5F   A9 12      LDA #$12      18
CA61   85 83      STA $83
CA63   20 07 D1   JSR $D107     Schreibkanal öffnen
CA66   20 D3 D1   JSR $D1D3     Drivenummer setzen
CA69   20 CB E1   JSR $E1CB
CA6C   20 9C E2   JSR $E29C
CA6F   A5 D6      LDA $D6
CA71   85 87      STA $87
CA73   A5 D5      LDA $D5
CA75   85 86      STA $86
CA77   A9 00      LDA #$00
CA79   85 88      STA $88
CA7B   85 D4      STA $D4
CA7D   85 D7      STA $D7
CA7F   68         PLA
CA80   85 D5      STA $D5
CA82   68         PLA
CA83   85 D6      STA $D6
CA85   4C 3B E3   JMP $E33B

****************************** R-Befehl,   'Rename'
CA88   20 20 C3   JSR $C320     Drivenummer aus Befehlszeile holen
CA8B   A5 E3      LDA $E3
CA8D   29 01      AND #$01
CA8F   85 E3      STA $E3       2. Laufwerknummer
CA91   C5 E2      CMP $E2       mit erster Laufwerknummer vergleichen
CA93   F0 02      BEQ $CA97     gleich ?
CA95   09 80      ORA #$80
CA97   85 E2      STA $E2
CA99   20 4F C4   JSR $C44F     Datei im Directory suchen
CA9C   20 E7 CA   JSR $CAE7     existieren die Namen ?
CA9F   A5 E3      LDA $E3
CAA1   29 01      AND #$01
CAA3   85 7F      STA $7F       Drivenummer
CAA5   A5 D9      LDA $D9
CAA7   85 81      STA $81       Sektornummer
CAA9   20 57 DE   JSR $DE57     liest Block aus Directory
CAAC   20 99 D5   JSR $D599     ok ?
CAAF   A5 DE      LDA $DE       Zeiger auf Directoryeintrag
CAB1   18         CLC
CAB2   69 03      ADC #$03      plus 3 gleich Zeiger auf Dateinamen
```

```
CAB4   20 C8 D4   JSR $D4C8      Pufferzeiger setzen
CAB7   20 93 DF   JSR $DF93      Nummer des Puffers holen
CABA   A8         TAY
CABB   AE 7A 02   LDX $027A
CABE   A9 10      LDA #$10       16 Zeichen
CAC0   20 6E C6   JSR $C66E      Namen in Puffer schreiben
CAC3   20 5E DE   JSR $DE5E      Block auf Diskette schreiben
CAC6   20 99 D5   JSR $D599      ok ?
CAC9   4C 94 C1   JMP $C194      fertig, Diskstatus bereitstellen

****************************** prüft ob Datei vorhanden
CACC   A5 E8      LDA $E8        Dateityp
CACE   29 07      AND #$07
CAD0   8D 4A 02   STA $024A      merken
CAD3   AE 78 02   LDX $0278
CAD6   CA         DEX
CAD7   EC 77 02   CPX $0277
CADA   90 0A      BCC $CAE6
CADC   BD 80 02   LDA $0280,X    Tracknummer
CADF   D0 F5      BNE $CAD6      ungleich null ?
CAE1   A9 62      LDA #$62
CAE3   4C C8 C1   JMP $C1C8      62, 'file not found'
CAE6   60         RTS

CAE7   20 CC CA   JSR $CACC      Datei mit altem Namen vorhanden ?
CAEA   BD 80 02   LDA $0280,X    Tracknummer der neuen Datei
CAED   F0 05      BEQ $CAF4      Datei gelöscht ?
CAEF   A9 63      LDA #$63
CAF1   4C C8 C1   JMP $C1C8      63, 'file exists'
CAF4   CA         DEX
CAF5   10 F3      BPL $CAEA
CAF7   60         RTS

****************************** M-Befehle,   'Memory'
CAF8   AD 01 02   LDA $0201      zweites Zeichen aus Puffer
CAFB   C9 2D      CMP #$2D       '-'
CAFD   D0 4C      BNE $CB4B
CAFF   AD 03 02   LDA $0203
CB02   85 6F      STA $6F        Adresse nach $6F/$70
CB04   AD 04 02   LDA $0204
CB07   85 70      STA $70
CB09   A0 00      LDY #$00
CB0B   AD 02 02   LDA $0202      3. Zeichen aus Puffer
CB0E   C9 52      CMP #$52       'R'
CB10   F0 0E      BEQ $CB20      zum Memory-Read
CB12   20 58 F2   JSR $F258      (RTS)
CB15   C9 57      CMP #$57       'W'
CB17   F0 37      BEQ $CB50      zum Memory-Write
CB19   C9 45      CMP #$45       'E'
CB1B   D0 2E      BNE $CB4B
CB1D   6C 6F 00   JMP ($006F)    Memory-Execute, Routine ausführen

****************************** M-R,  'Memory-Read'
CB20   B1 6F      LDA ($6F),Y    Byte lesen
CB22   85 85      STA $85
CB24   AD 74 02   LDA $0274      Länge der Befehlszeile
```

```
CB27   C9 06      CMP #$06      kleiner 6 ?
CB29   90 1A      BCC $CB45     ja
CB2B   AE 05 02   LDX $0205     Anzahl
CB2E   CA         DEX
CB2F   F0 14      BEQ $CB45     nur ein Byte ?
CB31   8A         TXA           Anzahl der Bytes
CB32   18         CLC
CB33   65 6F      ADC $6F       plus Startadresse
CB35   E6 6F      INC $6F
CB37   8D 49 02   STA $0249     Endezeiger
CB3A   A5 6F      LDA $6F
CB3C   85 A5      STA $A5       Pufferzeiger für Fehlermeldung
CB3E   A5 70      LDA $70       auf Startadresse für M-R setzen
CB40   85 A6      STA $A6
CB42   4C 43 D4   JMP $D443     Byte ausgeben

CB45   20 EB D0   JSR $D0EB     Lesekanal öffnen
CB48   4C 3A D4   JMP $D43A     Bytes ausgeben

CB4B   A9 31      LDA #$31
CB4D   4C C8 C1   JMP $C1C8     31, 'syntax error'

******************************  M-W,   'memory-write'
CB50   B9 06 02   LDA $0206,Y   Zeichen lesen
CB53   91 6F      STA ($6F),Y   und speichern
CB55   C8         INY
CB56   CC 05 02   CPY $0205     Anzahl der Zeichen
CB59   90 F5      BCC $CB50     schon alle Zeichen ?
CB5B   60         RTS

******************************  U-Befehl,   'User'
CB5C   AC 01 02   LDY $0201     zweites Zeichen
CB5F   C0 30      CPY #$30      '0'
CB61   D0 09      BNE $CB6C     nein
CB63   A9 EA      LDA #$EA
CB65   85 6B      STA $6B       Zeiger auf Tabelle der User-Adressen
CB67   A9 FF      LDA #$FF      $FFEA
CB69   85 6C      STA $6C
CB6B   60         RTS

CB6C   20 72 CB   JSR $CB72
CB6F   4C 94 C1   JMP $C194     fertig, Fehlermeldung bereit stellen

CB72   88         DEY
CB73   98         TYA
CB74   29 0F      AND #$0F      Nummer
CB76   0A         ASL  A        mal 2
CB77   A8         TAY
CB78   B1 6B      LDA ($6B),Y   als Zeiger in Tabelle
CB7A   85 75      STA $75
CB7C   C8         INY           Adresse nach $75/$76
CB7D   B1 6B      LDA ($6B),Y
CB7F   85 76      STA $76
CB81   6C 75 00   JMP ($0075)   Funktion ausführen

******************************  Direktzugriffskanal öffnen, '#'
```

```
CB84   AD 8E 02   LDA $028E     letzte Drivenummer
CB87   85 7F      STA $7F      Drivenummer
CB89   A5 83      LDA $83      Kanalnummer
CB8B   48         PHA
CB8C   20 3D C6   JSR $C63D    Laufwerk prüfen und evtl. initialisieren
CB8F   68         PLA
CB90   85 83      STA $83
CB92   AE 74 02   LDX $0274    Länge des Filenamens
CB95   CA         DEX
CB96   D0 0D      BNE $CBA5    größer eins ?
CB98   A9 01      LDA #$01
CB9A   20 E2 D1   JSR $D1E2    Kanal und Puffer belegen
CB9D   4C F1 CB   JMP $CBF1    Flags setzen, fertig

CBA0   A9 70      LDA #$70
CBA2   4C C8 C1   JMP $C1C8    70, 'no channel'
CBA5   A0 01      LDY #$01
CBA7   20 7C CC   JSR $CC7C    Puffernummer holen
CBAA   AE 85 02   LDX $0285    Puffernummer
CBAD   E0 05      CPX #$05     größer gleich 5 ?
CBAF   B0 EF      BCS $CBA0    70,'no channel'
CBB1   A9 00      LDA #$00
CBB3   85 6F      STA $6F
CBB5   85 70      STA $70
CBB7   38         SEC
CBB8   26 6F      ROL $6F      Puffer in Belegungsregister suchen
CBBA   26 70      ROL $70
CBBC   CA         DEX
CBBD   10 F9      BPL $CBB8
CBBF   A5 6F      LDA $6F
CBC1   2D 4F 02   AND $024F
CBC4   D0 DA      BNE $CBA0    Puffer belegt ?
CBC6   A5 70      LDA $70
CBC8   2D 50 02   AND $0250
CBCB   D0 D3      BNE $CBA0    Puffer belegt ?
CBCD   A5 6F      LDA $6F
CBCF   0D 4F 02   ORA $024F
CBD2   8D 4F 02   STA $024F
CBD5   A5 70      LDA $70      Puffer belegen
CBD7   0D 50 02   ORA $0250
CBDA   8D 50 02   STA $0250
CBDD   A9 00      LDA #$00
CBDF   20 E2 D1   JSR $D1E2    Kanal suchen und belegen
CBE2   A6 82      LDX $82      Kanalnummer
CBE4   AD 85 02   LDA $0285    Puffernummer
CBE7   95 A7      STA $A7,X
CBE9   AA         TAX
CBEA   A5 7F      LDA $7F      Drivenummer
CBEC   95 00      STA $00,X
CBEE   9D 5B 02   STA $025B,X
CBF1   A6 83      LDX $83      Sekundäradresse
CBF3   BD 2B 02   LDA $022B,X
CBF6   09 40      ORA #$40     READ und WRITE-Flag setzen
CBF8   9D 2B 02   STA $022B,X
CBFB   A4 82      LDY $82      Kanalnummer
CBFD   A9 FF      LDA #$FF
```

```
CBFF   99 44 02   STA $0244,Y   Endezeiger
CC02   A9 89      LDA #$89
CC04   99 F2 00   STA $00F2,Y   READ und WRITE-Flag setzen
CC07   B9 A7 00   LDA $00A7,Y   Puffernummer
CC0A   99 3E 02   STA $023E,Y
CC0D   0A         ASL  A        mal 2
CC0E   AA         TAX
CC0F   A9 01      LDA #$01
CC11   95 99      STA $99,X     Pufferzeiger auf eins
CC13   A9 0E      LDA #$0E
CC15   99 EC 00   STA $00EC,Y   Flag für Direktzugriff
CC18   4C 94 C1   JMP $C194     fertig

****************************** B-Befehle,   'Block'
CC1B   A0 00      LDY #$00
CC1D   A2 00      LDX #$00
CC1F   A9 2D      LDA #$2D      '-'
CC21   20 68 C2   JSR $C268     sucht Minuszeichen
CC24   D0 0A      BNE $CC30     gefunden ?
CC26   A9 31      LDA #$31
CC28   4C C8 C1   JMP $C1C8     31, 'syntax error'

CC2B   A9 30      LDA #$30
CC2D   4C C8 C1   JMP $C1C8     30, 'syntax error'

CC30   8A         TXA
CC31   D0 F8      BNE $CC2B     Komma, dann Fehler
CC33   A2 05      LDX #$05
CC35   B9 00 02   LDA $0200,Y   Zeichen aus Puffer
CC38   DD 5D CC   CMP $CC5D,X   mit 'AFRWEP' vergleichen
CC3B   F0 05      BEQ $CC42     gefunden ?
CC3D   CA         DEX
CC3E   10 F8      BPL $CC38     mit allen Zeichen vergleichen
CC40   30 E4      BMI $CC26     nicht gefunden, Fehler
CC42   8A         TXA
CC43   09 80      ORA #$80      Befehlsnummer, Bit 7 setzen
CC45   8D 2A 02   STA $022A
CC48   20 6F CC   JSR $CC6F     Parameter holen
CC4B   AD 2A 02   LDA $022A
CC4E   0A         ASL  A        Nummer mal 2
CC4F   AA         TAX           als Index
CC50   BD 64 CC   LDA $CC64,X   Adresse des Befehls Hi
CC53   85 70      STA $70
CC55   BD 63 CC   LDA $CC63,X   Adresse lo
CC58   85 6F      STA $6F
CC5A   6C 6F 00   JMP ($006F)   Sprung auf Befehl

****************************** Namen der verschiedenen Blockbefehle
CC5D   41 46 52 57 45 50        'AFRWEP'

****************************** Adressen der Blockbefehle
CC63   03 CD                    $CD03, B-A
CC65   F5 CC                    $CCF5, B-F
CC67   56 CD                    $CD56, B-R
CC69   73 CD                    $CD73, B-W
CC6B   A3 CD                    $CDA3, B-E
```

```
CC6D   BD CD                    $CDBD, B-P

****************************** Parameter für Block-Befehle holen
CC6F   A0 00      LDY #$00
CC71   A2 00      LDX #$00
CC73   A9 3A      LDA #$3A      ':'
CC75   20 68 C2   JSR $C268     Zeile bis Doppelpunkt testen
CC78   D0 02      BNE $CC7C     gefunden ?
CC7A   A0 03      LDY #$03      nein, ab 4. Zeichen beginnen
CC7C   B9 00 02   LDA $0200,Y   Trennzeichen suchen
CC7F   C9 20      CMP #$20      ' ' Blank
CC81   F0 08      BEQ $CC8B
CC83   C9 1D      CMP #$1D      Cursor right
CC85   F0 04      BEQ $CC8B
CC87   C9 2C      CMP #$2C      ',' Komma
CC89   D0 07      BNE $CC92
CC8B   C8         INY
CC8C   CC 74 02   CPY $0274     Zeilenende ?
CC8F   90 EB      BCC $CC7C
CC91   60         RTS

CC92   20 A1 CC   JSR $CCA1     nächsten Parameter übernehmen
CC95   EE 77 02   INC $0277     Parameterzähler erhöhen
CC98   AC 79 02   LDY $0279
CC9B   E0 04      CPX #$04      mit Maximalzahl vergleichen
CC9D   90 EC      BCC $CC8B     noch nicht überschritten ?
CC9F   B0 8A      BCS $CC2B     30, 'syntax error'
CCA1   A9 00      LDA #$00
CCA3   85 6F      STA $6F
CCA5   85 70      STA $70       Speicherbereich für Dezimalziffern löschen
CCA7   85 72      STA $72
CCA9   A2 FF      LDX #$FF
CCAB   B9 00 02   LDA $0200,Y   Zeichen aus Eingabepuffer holen
CCAE   C9 40      CMP #$40
CCB0   B0 18      BCS $CCCA     keine Ziffer ?
CCB2   C9 30      CMP #$30      '0'
CCB4   90 14      BCC $CCCA     keine Ziffer ?
CCB6   29 0F      AND #$0F      ASCII-Ziffer nach Hex wandeln
CCB8   48         PHA           und merken
CCB9   A5 70      LDA $70
CCBB   85 71      STA $71       Ziffern eins weiter schieben
CCBD   A5 6F      LDA $6F
CCBF   85 70      STA $70
CCC1   68         PLA
CCC2   85 6F      STA $6F       gelesene Zahl merken
CCC4   C8         INY           Zeiger in Eingabepuffer erhöhen
CCC5   CC 74 02   CPY $0274     Zeilenende erreicht ?
CCC7   90 E1      BCC $CCAB     nein
CCCA   8C 79 02   STY $0279     Zeiger merken
CCCD   18         CLC
CCCE   A9 00      LDA #$00
CCD0   E8         INX
CCD1   E0 03      CPX #$03
CCD3   B0 0F      BCS $CCE4     Umrechnung der Hexziffern in ein Byte
CCD5   B4 6F      LDY $6F,X
CCD7   88         DEY
```

```
CCD8   30 F6      BMI $CCD0
CCDA   7D F2 CC   ADC $CCF2,X   dezimale Wertigkeit addieren
CCDD   90 F8      BCC $CCD7
CCDF   18         CLC
CCE0   E6 72      INC $72
CCE2   D0 F3      BNE $CCD7
CCE4   48         PHA
CCE5   AE 77 02   LDX $0277     Zähler für Parameter
CCE8   A5 72      LDA $72
CCEA   9D 80 02   STA $0280,X   Hi-Byte
CCED   68         PLA
CCEE   9D 85 02   STA $0285,X   Lo-Byte
CCF1   60         RTS

******************************  Dezimalwerte
CCF2   01 0A 64                 1, 10, 100

******************************  B-F Befehl,   'Block free'
CCF5   20 F5 CD   JSR $CDF5     Track, Sektor und Drivenummer holen
CCF8   20 5F EF   JSR $EF5F     Block freigeben
CCFB   4C 94 C1   JMP $C194     fertig, Fehlermeldung bereit stelllen

******************************
CCFE   A9 01      LDA #$01
CD00   8D F9 02   STA $02F9

******************************  B-A Befehl,   'Block allocate'
CD03   20 F5 CD   JSR $CDF5     Track, Sektor und Drivenummer holen
CD06   A5 81      LDA $81       Sektor
CD08   48         PHA           merken
CD09   20 FA F1   JSR $F1FA     sucht Block in BAM
CD0C   F0 0B      BEQ $CD19     Block schon belegt ?
CD0E   68         PLA           gewünschter Sektor
CD0F   C5 81      CMP $81       gleich nächster freier Sektor ?
CD11   D0 19      BNE $CD2C     nein
CD13   20 90 EF   JSR $EF90     Block in BAM belegen
CD16   4C 94 C1   JMP $C194     fertig

CD19   68         PLA
CD1A   A9 00      LDA #$00
CD1C   85 81      STA $81       Sektor 0
CD1E   E6 80      INC $80       nächster Track
CD20   A5 80      LDA $80       Tracknummer
CD22   CD D7 FE   CMP $FED7     36, letzte Tracknummer + 1
CD25   B0 0A      BCS $CD31     größer oder gleich, dann 'no block'
CD27   20 FA F1   JSR $F1FA     freien Block im nächsten Track suchen
CD2A   F0 EE      BEQ $CD1A     nicht gefunden, nächsten Track prüfen
CD2C   A9 65      LDA #$65
CD2E   20 45 E6   JSR $E645     65, 'no block' nächster freier Block
CD31   A9 65      LDA #$65
CD33   20 C8 C1   JSR $C1C8     65, 'no block' kein Block mehr frei

******************************
CD36   20 F2 CD   JSR $CDF2     Kanal öffnen, Parameter setzen
CD39   4C 60 D4   JMP $D460     Block von Diskette lesen
```

```
******************************  Byte aus Puffer holen
CD3C   20 2F D1   JSR $D12F     Zeiger auf Puffer setzen
CD3F   A1 99      LDA ($99,X)   Byte holen
CD41   60         RTS

******************************  Block von Diskette lesen
CD42   20 36 CD   JSR $CD36     Kanal öffnen, Block lesen
CD45   A9 00      LDA #$00
CD47   20 C8 D4   JSR $D4C8     Pufferzeiger auf Null setzen
CD4A   20 3C CD   JSR $CD3C     ein Byte aus Puffer holen
CD4D   99 44 02   STA $0244,Y
CD50   A9 89      LDA #$89      Schreib- und Leseflag setzen
CD52   99 F2 00   STA $00F2,Y
CD55   60         RTS

******************************  B-R Befehl,   'Block Read'
CD56   20 42 CD   JSR $CD42     Block von Diskette lesen
CD59   20 EC D3   JSR $D3EC     Byte aus Puffer bereitstellen
CD5C   4C 94 C1   JMP $C194     Fehlermeldung bereitstellen

******************************  U1 Befehl, Ersatz für 'Block-Read'
CD5F   20 6F CC   JSR $CC6F     Parameter des Befehls holen
CD62   20 42 CD   JSR $CD42     Block von Diskette lesen
CD65   B9 44 02   LDA $0244,Y   Endezeiger
CD68   99 3E 02   STA $023E,Y   als Datenbyte speichern
CD6B   A9 FF      LDA #$FF
CD6D   99 44 02   STA $0244,Y   Endezeiger auf $FF
CD70   4C 94 C1   JMP $C194     fertig, Fehlermeldung bereit stellen

******************************  B-W Befehl,   'Block Write'
CD73   20 F2 CD   JSR $CDF2     Kanal öffnen
CD76   20 E8 D4   JSR $D4E8     Pufferzeiger setzen
CD79   A8         TAY
CD7A   88         DEY
CD7B   C9 02      CMP #$02      Pufferzeiger lo kleiner 2
CD7D   B0 02      BCS $CD81     nein
CD7F   A0 01      LDY #$01
CD81   A9 00      LDA #$00
CD83   20 C8 D4   JSR $D4C8     Pufferzeiger auf null
CD86   98         TYA
CD87   20 F1 CF   JSR $CFF1     Byte in Puffer schreiben
CD8A   8A         TXA
CD8B   48         PHA
CD8C   20 64 D4   JSR $D464     Block auf Diskette schreiben
CD8F   68         PLA
CD90   AA         TAX
CD91   20 EE D3   JSR $D3EE     Byte aus Puffer holen
CD94   4C 94 C1   JMP $C194     fertig, Fehlermeldung

******************************  U2, Ersatz für 'Block write'
CD97   20 6F CC   JSR $CC6F     Parameter des Befehls holen
CD9A   20 F2 CD   JSR $CDF2     Kanal öffnen
CD9D   20 64 D4   JSR $D464     und Block auf Diskette schreiben
CDA0   4C 94 C1   JMP $C194     fertig

******************************  'B-E' Befehl, 'Block execute'
```

```
CDA3   20 58 F2   JSR $F258      (RTS)
CDA6   20 36 CD   JSR $CD36      Kanal öffnen und Block einlesen
CDA9   A9 00      LDA #$00
CDAB   85 6F      STA $6F         Adresse low
CDAD   A6 F9      LDX $F9         Puffernummer
CDAF   BD E0 FE   LDA $FEE0,X     Pufferadresse high
CDB2   85 70      STA $70
CDB4   20 BA CD   JSR $CDBA       Routine ausführen
CDB7   4C 94 C1   JMP $C194       fertig
CDBA   6C 6F 00   JMP ($006F)     Sprung auf Routine

******************************  'B-P' Befehl, 'Block pointer'
CDBD   20 D2 CD   JSR $CDD2       Kanal öffnen, Puffernummer holen
CDC0   A5 F9      LDA $F9         Puffernummer
CDC2   0A         ASL  A          * 2
CDC3   AA         TAX             als Index
CDC4   AD 86 02   LDA $0286       Pointerwert
CDC7   95 99      STA $99,X       als Pufferzeiger abspeichern
CDC9   20 2F D1   JSR $D12F       ein Byte aus Puffer
CDCC   20 EE D3   JSR $D3EE       zur Ausgabe bereitstellen
CDCF   4C 94 C1   JMP $C194       fertig

******************************  Kanal öffnen
CDD2   A6 D3      LDX $D3
CDD4   E6 D3      INC $D3
CDD6   BD 85 02   LDA $0285,X     Puffernummer
CDD9   A8         TAY
CDDA   88         DEY
CDDB   88         DEY
CDDC   C0 0C      CPY #$0C        Puffernummer kleiner 14 ?
CDDE   90 05      BCC $CDE5       ja
CDE0   A9 70      LDA #$70
CDE2   4C C8 C1   JMP $C1C8       70, 'no channel'

CDE5   85 83      STA $83         Sekundäradresse
CDE7   20 EB D0   JSR $D0EB       Kanal öffnen
CDEA   B0 F4      BCS $CDE0       schon belegt, dann 70, 'no channel'
CDEC   20 93 DF   JSR $DF93       Puffernummer
CDEF   85 F9      STA $F9         setzen
CDF1   60         RTS

******************************
CDF2   20 D2 CD   JSR $CDD2       Puffernummer prüfen und Kanal öffnen
CDF5   A6 D3      LDX $D3         Kanalnummer
CDF7   BD 85 02   LDA $0285,X     Pufferadresse
CDFA   29 01      AND #$01
CDFC   85 7F      STA $7F         Drivenummer
CDFE   BD 87 02   LDA $0287,X
CE01   85 81      STA $81         Sektor
CE03   BD 86 02   LDA $0286,X
CE06   85 80      STA $80         Track
CE08   20 5F D5   JSR $D55F       Track und Sektor ok ?
CE0B   4C 00 C1   JMP $C100       LED einschalten

******************************  Pointer für REL-Datei setzen
CE0E   20 2C CE   JSR $CE2C       Recordnummer * Recordlänge
```

```
CE11   20 6E CE   JSR $CE6E     durch 254 gleich Datenblocknummer
CE14   A5 90      LDA $90       Rest der Division gleich Zeiger in Datenblock
CE16   85 D7      STA $D7       Datenzeiger
CE18   20 71 CE   JSR $CE71     durch 120 gleich Side Sektornummer
CE1B   E6 D7      INC $D7
CE1D   E6 D7      INC $D7       Datenzeiger plus 2 (Track/Sektor-Zeiger!)
CE1F   A5 8B      LDA $8B       Ergebnis der Division
CE21   85 D5      STA $D5       gleich Side Sektornummer
CE23   A5 90      LDA $90       Rest der Division
CE25   0A         ASL  A        mal 2
CE26   18         CLC
CE27   69 10      ADC #$10      plus 16
CE29   85 D6      STA $D6       gleich Zeiger in Side-Sektor auf Datenblock
CE2B   60         RTS

******************************
CE2C   20 D9 CE   JSR $CED9     Arbeitsspeicher löschen
CE2F   85 92      STA $92
CE31   A6 82      LDX $82       Kanalnummer
CE33   B5 B5      LDA $B5,X     Recordnummer lo
CE35   85 90      STA $90
CE37   B5 BB      LDA $BB,X     Recordnummer hi
CE39   85 91      STA $91
CE3B   D0 04      BNE $CE41
CE3D   A5 90      LDA $90
CE3F   F0 0B      BEQ $CE4C     Recordnummer ungleich 0 ?
CE41   A5 90      LDA $90
CE43   38         SEC
CE44   E9 01      SBC #$01      dann eins abziehen
CE46   85 90      STA $90
CE48   B0 02      BCS $CE4C
CE4A   C6 91      DEC $91
CE4C   B5 C7      LDA $C7,X     Recordlänge
CE4E   85 6F      STA $6F
CE50   46 6F      LSR $6F
CE52   90 03      BCC $CE57
CE54   20 ED CE   JSR $CEED     Recordnummer * Recordlänge
CE57   20 E5 CE   JSR $CEE5     Register linksverschieben
CE5A   A5 6F      LDA $6F
CE5C   D0 F2      BNE $CE50
CE5E   A5 D4      LDA $D4
CE60   18         CLC
CE61   65 8B      ADC $8B
CE63   85 8B      STA $8B
CE65   90 06      BCC $CE6D     Ergebnis in $8B/$8C/$8D
CE67   E6 8C      INC $8C
CE69   D0 02      BNE $CE6D
CE6B   E6 8D      INC $8D
CE6D   60         RTS

******************************   Division durch 254, Datenblocknummer berechnen
CE6E   A9 FE      LDA #$FE      254
CE70   2C         .BYTE $2C
******************************   Divison durch 120, Side Sektornummer berechnen
CE71   A9 78      LDA #$78      120
CE73   85 6F      STA $6F       Divisor
```

```
CE75   A2 03      LDX #$03
CE77   B5 8F      LDA $8F,X
CE79   48         PHA
CE7A   B5 8A      LDA $8A,X
CE7C   95 8F      STA $8F,X
CE7E   68         PLA
CE7F   95 8A      STA $8A,X
CE81   CA         DEX
CE82   D0 F3      BNE $CE77
CE84   20 D9 CE   JSR $CED9     Arbeitsspeicher löschen
CE87   A2 00      LDX #$00
CE89   B5 90      LDA $90,X
CE8B   95 8F      STA $8F,X
CE8D   E8         INX
CE8E   E0 04      CPX #$04
CE90   90 F7      BCC $CE89
CE92   A9 00      LDA #$00
CE94   85 92      STA $92
CE96   24 6F      BIT $6F
CE98   30 09      BMI $CEA3
CE9A   06 8F      ASL $8F
CE9C   08         PHP
CE9D   46 8F      LSR $8F
CE9F   28         PLP
CEA0   20 E6 CE   JSR $CEE6     Register 1 linksverschieben
CEA3   20 ED CE   JSR $CEED     Register 0 zu Register 1 addieren
CEA6   20 E5 CE   JSR $CEE5     Register 1 linksverschieben
CEA9   24 6F      BIT $6F
CEAB   30 03      BMI $CEB0
CEAD   20 E2 CE   JSR $CEE2     Register 1 zweimal linksverschieben
CEB0   A5 8F      LDA $8F
CEB2   18         CLC
CEB3   65 90      ADC $90
CEB5   85 90      STA $90
CEB7   90 06      BCC $CEBF
CEB9   E6 91      INC $91
CEBB   D0 02      BNE $CEBF
CEBD   E6 92      INC $92
CEBF   A5 92      LDA $92
CEC1   05 91      ORA $91
CEC3   D0 C2      BNE $CE87
CEC5   A5 90      LDA $90
CEC7   38         SEC
CEC8   E5 6F      SBC $6F       Quotient in $8B/$8C/$8D
CECA   90 0C      BCC $CED8
CECC   E6 8B      INC $8B
CECE   D0 06      BNE $CED6
CED0   E6 8C      INC $8C
CED2   D0 02      BNE $CED6
CED4   E6 8D      INC $8D
CED6   85 90      STA $90       Rest in $90
CED8   60         RTS

****************************** Arbeitsspeicher löschen
CED9   A9 00      LDA #$00
CEDB   85 8B      STA $8B
```

```
CEDD   85 8C      STA $8C
CEDF   85 8D      STA $8D
CEE1   60         RTS

****************************  3-Byte-Register zweimal linksverschieben
CEE2   20 E5 CE   JSR $CEE5

****************************  3-Byte-Register einmal linksverschieben
CEE5   18         CLC
CEE6   26 90      ROL $90
CEE8   26 91      ROL $91
CEEA   26 92      ROL $92
CEEC   60         RTS

****************************
CEED   18         CLC
CEEE   A2 FD      LDX #$FD
CEF0   B5 8E      LDA $8E,X       Register $90/$91/$92
CEF2   75 93      ADC $93,X       zu Register $8B/$8C/$8D addieren
CEF4   95 8E      STA $8E,X
CEF6   E8         INX
CEF7   D0 F7      BNE $CEF0
CEF9   60         RTS

CEFA   A2 00      LDX #$00
CEFC   8A         TXA
CEFD   95 FA      STA $FA,X
CEFF   E8         INX
CF00   E0 04      CPX #$04
CF02   D0 F8      BNE $CEFC
CF04   A9 06      LDA #$06
CF06   95 FA      STA $FA,X
CF08   60         RTS

CF09   A0 04      LDY #$04
CF0B   A6 82      LDX $82         Kanalnummer
CF0D   B9 FA 00   LDA $00FA,Y
CF10   96 FA      STX $FA,Y
CF12   C5 82      CMP $82         Kanalnummer
CF14   F0 07      BEQ $CF1D
CF16   88         DEY
CF17   30 E1      BMI $CEFA
CF19   AA         TAX
CF1A   4C 0D CF   JMP $CF0D
CF1D   60         RTS

CF1E   20 09 CF   JSR $CF09
CF21   20 B7 DF   JSR $DFB7
CF24   D0 46      BNE $CF6C
CF26   20 D3 D1   JSR $D1D3       Drivenummer setzen
CF29   20 8E D2   JSR $D28E
CF2C   30 48      BMI $CF76
CF2E   20 C2 DF   JSR $DFC2
CF31   A5 80      LDA $80         Track
CF33   48         PHA
CF34   A5 81      LDA $81         Sektor
```

```
CF36   48         PHA
CF37   A9 01      LDA #$01
CF39   20 F6 D4   JSR $D4F6     Byte 1 aus Puffer holen
CF3C   85 81      STA $81       Sektor
CF3E   A9 00      LDA #$00
CF40   20 F6 D4   JSR $D4F6     Byte 0 aus Puffer holen
CF43   85 80      STA $80       Track
CF45   F0 1F      BEQ $CF66
CF47   20 25 D1   JSR $D125     Dateityp prüfen
CF4A   F0 0B      BEQ $CF57     Rel-Datei ?
CF4C   20 AB DD   JSR $DDAB
CF4F   D0 06      BNE $CF57
CF51   20 8C CF   JSR $CF8C
CF54   4C 5D CF   JMP $CF5D

CF57   20 8C CF   JSR $CF8C
CF5A   20 57 DE   JSR $DE57
CF5D   68         PLA
CF5E   85 81      STA $81       Sektor-
CF60   68         PLA
CF61   85 80      STA $80       und Tracknummer zurückholen
CF63   4C 6F CF   JMP $CF6F

CF66   68         PLA
CF67   85 81      STA $81       Sektor-
CF69   68         PLA
CF6A   85 80      STA $80       und Tracknummer zurückholen
CF6C   20 8C CF   JSR $CF8C
CF6F   20 93 DF   JSR $DF93
CF72   AA         TAX
CF73   4C 99 D5   JMP $D599     und prüfen

CF76   A9 70      LDA #$70
CF78   4C C8 C1   JMP $C1C8     70, 'no channel'

CF7B   20 09 CF   JSR $CF09
CF7E   20 B7 DF   JSR $DFB7
CF81   D0 08      BNE $CF8B
CF83   20 8E D2   JSR $D28E
CF86   30 EE      BMI $CF76
CF88   20 C2 DF   JSR $DFC2
CF8B   60         RTS

******************************  Puffer wechseln
CF8C   A6 82      LDX $82       Kanalnummer
CF8E   B5 A7      LDA $A7,X
CF90   49 80      EOR #$80
CF92   95 A7      STA $A7,X
CF94   B5 AE      LDA $AE,X     Bit 7 in Tabelle umdrehen
CF96   49 80      EOR #$80
CF98   95 AE      STA $AE,X
CF9A   60         RTS

******************************  Datenbyte in Puffer schreiben
CF9B   A2 12      LDX #$12      Kanal 18
CF9D   86 83      STX $83
```

```
CF9F   20 07 D1   JSR $D107     Schreibkanal öffnen
CFA2   20 00 C1   JSR $C100     LED einschalten
CFA5   20 25 D1   JSR $D125     Dateityp prüfen
CFA8   90 05      BCC $CFAF     keine Rel-Datei
CFAA   A9 20      LDA #$20
CFAC   20 9D DD   JSR $DD9D     Puffer wechseln
CFAF   A5 83      LDA $83       Sekundäradresse
CFB1   C9 0F      CMP #$0F      15 ?
CFB3   F0 23      BEQ $CFD8     ja
CFB5   D0 08      BNE $CFBF     nein

CFB7   A5 84      LDA $84       Sekundäradresse
CFB9   29 8F      AND #$8F
CFBB   C9 0F      CMP #$0F      größer gleich 15 ?
CFBD   B0 19      BCS $CFD8     dann Eingabepuffer
CFBF   20 25 D1   JSR $D125     Dateityp prüfen
CFC2   B0 05      BCS $CFC9     Rel-Datei oder Direktzugriff ?
CFC4   A5 85      LDA $85       Datenbyte
CFC6   4C 9D D1   JMP $D19D     in Puffer schreiben

CFC9   D0 03      BNE $CFCE     Direktzugriffsdatei ?
CFCB   4C AB E0   JMP $E0AB     Datenbyte in Rel-Datei schreiben

CFCE   A5 85      LDA $85
CFD0   20 F1 CF   JSR $CFF1     Datenbyte in Puffer schreiben
CFD3   A4 82      LDY $82       Kanalnummer
CFD5   4C EE D3   JMP $D3EE     nächstes Byte zur Ausgabe bereitstellen

CFD8   A9 04      LDA #$04      Kanal 4
CFDA   85 82      STA $82       entspricht Eingabepuffer
CFDC   20 E8 D4   JSR $D4E8     Pufferzeiger setzen
CFDF   C9 2A      CMP #$2A      42
CFE1   F0 05      BEQ $CFE8     Pufferende ?
CFE3   A5 85      LDA $85
CFE5   20 F1 CF   JSR $CFF1     Datenbyte in Puffer schreiben
CFE8   A5 F8      LDA $F8       Endflag gesetzt ?
CFEA   F0 01      BEQ $CFED     ja
CFEC   60         RTS

CFED   EE 55 02   INC $0255     Kommandoflag setzen
CFF0   60         RTS

******************************  Datenbyte in Puffer schreiben
CFF1   48         PHA           Datenbyte merken
CFF2   20 93 DF   JSR $DF93     Puffernummer holen
CFF5   10 06      BPL $CFFD     Puffer zugeordnet ?
CFF7   68         PLA
CFF8   A9 61      LDA #$61
CFFA   4C C8 C1   JMP $C1C8     61, 'file not open'
CFFD   0A         ASL  A        Puffernummer mal 2
CFFE   AA         TAX           als Index
CFFF   68         PLA           Datenbyte
D000   81 99      STA ($99,X)   in Puffer schreiben
D002   F6 99      INC $99,X     Pufferzeiger erhöhen
D004   60         RTS
```

```
******************************  I-Befehl, Initialisieren
D005   20 D1 C1   JSR $C1D1     Drivenummer suchen
D008   20 42 D0   JSR $D042     BAM laden
D00B   4C 94 C1   JMP $C194     Diskstatus bereitstellen

******************************
D00E   20 0F F1   JSR $F10F
D011   A8         TAY
D012   B6 A7      LDX $A7,Y
D014   E0 FF      CPX #$FF
D016   D0 14      BNE $D02C
D018   48         PHA
D019   20 8E D2   JSR $D28E
D01C   AA         TAX
D01D   10 05      BPL $D024
D01F   A9 70      LDA #$70
D021   20 48 E6   JSR $E648     70, 'no channel'
D024   68         PLA
D025   A8         TAY
D026   8A         TXA
D027   09 80      ORA #$80
D029   99 A7 00   STA $00A7,Y
D02C   8A         TXA
D02D   29 0F      AND #$0F
D02F   85 F9      STA $F9
D031   A2 00      LDX #$00
D033   86 81      STX $81       Sektor 0
D035   AE 85 FE   LDX $FE85     18
D038   86 80      STX $80       Track 18
D03A   20 D3 D6   JSR $D6D3     Parameter an Disk-Controller übergeben
D03D   A9 B0      LDA #$B0      Befehlskode 'Block Header lesen'
D03F   4C 8C D5   JMP $D58C     an Disk-Controller übergeben

******************************  BAM laden
D042   20 D1 F0   JSR $F0D1
D045   20 13 D3   JSR $D313
D048   20 0E D0   JSR $D00E     Block lesen
D04B   A6 7F      LDX $7F       Drivenummer
D04D   A9 00      LDA #$00
D04F   9D 51 02   STA $0251,X   Flag für 'BAM geändert' rücksetzen
D052   8A         TXA
D053   0A         ASL  A
D054   AA         TAX
D055   A5 16      LDA $16
D057   95 12      STA $12,X
D059   A5 17      LDA $17       ID speichern
D05B   95 13      STA $13,X
D05D   20 86 D5   JSR $D586
D060   A5 F9      LDA $F9       Puffernummer
D062   0A         ASL  A
D063   AA         TAX
D064   A9 02      LDA #$02      Pufferzeiger auf $200
D066   95 99      STA $99,X
D068   A1 99      LDA ($99,X)   Zeichen aus Puffer holen
D06A   A6 7F      LDX $7F       Drivenummer
D06C   9D 01 01   STA $0101,X
```

```
D06F   A9 00      LDA #$00
D071   95 1C      STA $1C,X      Flag für Write Protect
D073   95 FF      STA $FF,X      Flag für Lesefehler

****************************  Blocks free berechnen
D075   20 3A EF   JSR $EF3A      Pufferadresse nach $6D/$6E
D078   A0 04      LDY #$04       bei Position 4 beginnen
D07A   A9 00      LDA #$00
D07C   AA         TAX
D07D   18         CLC
D07E   71 6D      ADC ($6D),Y    Anzahl freie Blocks pro Track addieren
D080   90 01      BCC $D083
D082   E8         INX            X als Hi-Byte
D083   C8         INY
D084   C8         INY            plus 4
D085   C8         INY
D086   C8         INY
D087   C0 48      CPY #$48       Track 18 ?
D089   F0 F8      BEQ $D083      dann übergehen
D08B   C0 90      CPY #$90       letzte Tracknummer ?
D08D   D0 EE      BNE $D07D      nein
D08F   48         PHA            Lo-Byte
D090   8A         TXA            Hi-Byte
D091   A6 7F      LDX $7F        Drivenummer
D093   9D FC 02   STA $02FC,X    Hi-Byte nach $2FC
D096   68         PLA            Lo-Byte
D097   9D FA 02   STA $02FA,X    nach $2FA
D09A   60         RTS

****************************
D09B   20 D0 D6   JSR $D6D0      Parameter an Disk-Controller
D09E   20 C3 D0   JSR $D0C3      Block lesen
D0A1   20 99 D5   JSR $D599      ok ?
D0A4   20 37 D1   JSR $D137      Byte aus Puffer holen
D0A7   85 80      STA $80        Track
D0A9   20 37 D1   JSR $D137      nächstes Byte aus Puffer
D0AC   85 81      STA $81        Sektor
D0AE   60         RTS

D0AF   20 9B D0   JSR $D09B
D0B2   A5 80      LDA $80        Track
D0B4   D0 01      BNE $D0B7
D0B6   60         RTS
D0B7   20 1E CF   JSR $CF1E      Puffer wechseln
D0BA   20 D0 D6   JSR $D6D0      Parameter an Disk-Controller
D0BD   20 C3 D0   JSR $D0C3      Block lesen
D0C0   4C 1E CF   JMP $CF1E      Puffer wechseln

****************************  Block lesen
D0C3   A9 80      LDA #$80       Kode für 'Lesen'
D0C5   D0 02      BNE $D0C9

****************************  Block schreiben
D0C7   A9 90      LDA #$90       Kode für 'Schreiben'
D0C9   8D 4D 02   STA $024D      merken
D0CC   20 93 DF   JSR $DF93      Puffernummer holen
```

```
D0CF   AA         TAX
D0D0   20 06 D5   JSR $D506     Track/Sektor holen, Block lesen/schreiben
D0D3   8A         TXA
D0D4   48         PHA
D0D5   0A         ASL  A       Pufferzeiger mal 2
D0D6   AA         TAX
D0D7   A9 00      LDA #$00
D0D9   95 99      STA $99,X    Zeiger in Puffer auf null
D0DB   20 25 D1   JSR $D125    Dateityp holen
D0DE   C9 04      CMP #$04     Rel-Datei oder Direktzugriff ?
D0E0   B0 06      BCS $D0E8    ja
D0E2   F6 B5      INC $B5,X
D0E4   D0 02      BNE $D0E8    Blockzähler erhöhen
D0E6   F6 BB      INC $BB,X
D0E8   68         PLA
D0E9   AA         TAX
D0EA   60         RTS

****************************** Kanal zum Lesen öffnen
D0EB   A5 83      LDA $83      Sekundäradresse
D0ED   C9 13      CMP #$13     19
D0EF   90 02      BCC $D0F3    kleiner ?
D0F1   29 0F      AND #$0F
D0F3   C9 0F      CMP #$0F
D0F5   D0 02      BNE $D0F9
D0F7   A9 10      LDA #$10     16
D0F9   AA         TAX
D0FA   38         SEC
D0FB   BD 2B 02   LDA $022B,X
D0FE   30 06      BMI $D106
D100   29 0F      AND #$0F
D102   85 82      STA $82
D104   AA         TAX
D105   18         CLC          Flag für ok
D106   60         RTS

****************************** Kanal zum Schreiben öffnen
D107   A5 83      LDA $83      Sekundäradresse
D109   C9 13      CMP #$13     19
D10B   90 02      BCC $D10F    kleiner ?
D10D   29 0F      AND #$0F
D10F   AA         TAX
D110   BD 2B 02   LDA $022B,X  Kanalnummer
D113   A8         TAY
D114   0A         ASL  A
D115   90 0A      BCC $D121
D117   30 0A      BMI $D123
D119   98         TYA
D11A   29 0F      AND #$0F
D11C   85 82      STA $82
D11E   AA         TAX
D11F   18         CLC          Flag für ok
D120   60         RTS

D121   30 F6      BMI $D119
D123   38         SEC          Flag für Kanal belegt
```

```
D124   60         RTS

****************************** Auf Filtyp 'REL' prüfen
D125   A6 82      LDX $82
D127   B5 EC      LDA $EC,X
D129   4A         LSR  A
D12A   29 07      AND #$07
D12C   C9 04      CMP #$04       'REL' ?
D12E   60         RTS

****************************** Puffer- und Kanalnummer holen
D12F   20 93 DF   JSR $DF93      Puffernummer holen
D132   0A         ASL  A
D133   AA         TAX
D134   A4 82      LDY $82
D136   60         RTS

****************************** ein Byte aus Puffer holen
D137   20 2F D1   JSR $D12F      Puffer- und Kanalnummer holen
D13A   B9 44 02   LDA $0244,Y    Endezeiger
D13D   F0 12      BEQ $D151
D13F   A1 99      LDA ($99,X)    Byte aus Puffer holen
D141   48         PHA
D142   B5 99      LDA $99,X      Pufferzeiger
D144   D9 44 02   CMP $0244,Y    gleich Endezeiger ?
D147   D0 04      BNE $D14D      nein
D149   A9 FF      LDA #$FF
D14B   95 99      STA $99,X      Pufferzeiger auf -1
D14D   68         PLA            Datenbyte
D14E   F6 99      INC $99,X      Pufferzeiger erhöhen
D150   60         RTS
D151   A1 99      LDA ($99,X)    Zeichen aus Puffer holen
D153   F6 99      INC $99,X      Pufferzeiger erhöhen
D155   60         RTS

****************************** Byte holen und evtl. nächsten Block lesen
D156   20 37 D1   JSR $D137      Byte aus Puffer holen
D159   D0 36      BNE $D191      nicht das letzte Zeichen ?
D15B   85 85      STA $85        Datenbyte merken
D15D   B9 44 02   LDA $0244,Y    Endezeiger
D160   F0 08      BEQ $D16A      ja
D162   A9 80      LDA #$80
D164   99 F2 00   STA $00F2,Y    READ-Flag
D167   A5 85      LDA $85        Datenbyte
D169   60         RTS

D16A   20 1E CF   JSR $CF1E      Puffer wechseln und nächsten Block lesen
D16D   A9 00      LDA #$00
D16F   20 C8 D4   JSR $D4C8      Puffezeiger auf null setzen
D172   20 37 D1   JSR $D137      erstes Byte aus Puffer holen
D175   C9 00      CMP #$00       Tracknummer Null ?
D177   F0 19      BEQ $D192      ja, dann letzter Block
D179   85 80      STA $80        Tracknummer merken
D17B   20 37 D1   JSR $D137      nächstes Byte holen
D17E   85 81      STA $81        als Folgesektor merken
D180   20 1E CF   JSR $CF1E      Puffer wechseln und nächsten Block lesen
```

```
D183   20 D3 D1   JSR $D1D3      Drivenummer merken
D186   20 D0 D6   JSR $D6D0      Parameter an Disk-Controller
D189   20 C3 D0   JSR $D0C3      Lesebefehl übergeben
D18C   20 1E CF   JSR $CF1E      Puffer wechseln und nächstem Block lesen
D18F   A5 85      LDA $85        Datenbyte zurückholen
D191   60         RTS

D192   20 37 D1   JSR $D137      nächstes Byte aus Puffer holen
D195   A4 82      LDY $82
D197   99 44 02   STA $0244,Y    als Endezeiger merken
D19A   A5 85      LDA $85        Datenbyte zurückholen
D19C   60         RTS

****************************** Byte in Puffer und Block schreiben
D19D   20 F1 CF   JSR $CFF1      Byte in Puffer
D1A0   F0 01      BEQ $D1A3      Puffer voll ?
D1A2   60         RTS

D1A3   20 D3 D1   JSR $D1D3      Drivenummer holen
D1A6   20 1E F1   JSR $F11E      freien Block in BAM suchen
D1A9   A9 00      LDA #$00
D1AB   20 C8 D4   JSR $D4C8      Pufferzeiger auf Null
D1AE   A5 80      LDA $80
D1B0   20 F1 CF   JSR $CFF1      Tracknummer als erstes Byte
D1B3   A5 81      LDA $81
D1B5   20 F1 CF   JSR $CFF1      Sektornummer als zweites Byte
D1B8   20 C7 D0   JSR $D0C7      Block schreiben
D1BB   20 1E CF   JSR $CF1E      Puffer wechseln
D1BE   20 D0 D6   JSR $D6D0      Parameter an Disk-Controller
D1C1   A9 02      LDA #$02
D1C3   4C C8 D4   JMP $D4C8      Pufferzeiger auf 2

****************************** Pufferzeiger erhöhen
D1C6   85 6F      STA $6F
D1C8   20 E8 D4   JSR $D4E8      Pufferzeiger holen
D1CB   18         CLC
D1CC   65 6F      ADC $6F
D1CE   95 99      STA $99,X      und erhöhen
D1D0   85 94      STA $94
D1D2   60         RTS

****************************** Drivenummer holen
D1D3   20 93 DF   JSR $DF93      Puffernummer holen
D1D6   AA         TAX
D1D7   BD 5B 02   LDA $025B,X
D1DA   29 01      AND #$01       Drivenummer isolieren
D1DC   85 7F      STA $7F        und merken
D1DE   60         RTS

****************************** Schreibkanal und Puffer suchen
D1DF   38         SEC            Flag für Schreiben
D1E0   B0 01      BCS $D1E3

****************************** Lesekanal und Puffer suchen
D1E2   18         CLC            Flag für Lesen
D1E3   08         PHP            merken
```

```
D1E4   85 6F      STA $6F       Anzahl der Puffer
D1E6   20 27 D2   JSR $D227     Kanal schließen
D1E9   20 7F D3   JSR $D37F     freien Kanal belegen
D1EC   85 82      STA $82       Kanalnummer
D1EE   A6 83      LDX $83       Sekundäradresse
D1F0   28         PLP
D1F1   90 02      BCC $D1F5     Lesekanal ?
D1F3   09 80      ORA #$80      Flag für Schreiben
D1F5   9D 2B 02   STA $022B,X   setzen
D1F8   29 3F      AND #$3F
D1FA   A8         TAY
D1FB   A9 FF      LDA #$FF      Defaultwert
D1FD   99 A7 00   STA $00A7,Y
D200   99 AE 00   STA $00AE,Y   in Zuordungstabellen schreiben
D203   99 CD 00   STA $00CD,Y
D206   C6 6F      DEC $6F       Zahl der Puffer erniedrigen
D208   30 1C      BMI $D226     schon fertig ?
D20A   20 8E D2   JSR $D28E     Puffer suchen
D20D   10 08      BPL $D217     gefunden ?
D20F   20 5A D2   JSR $D25A     Flags in Tabelle löschen
D212   A9 70      LDA #$70
D214   4C C8 C1   JMP $C1C8     70, 'no channel'
D217   99 A7 00   STA $00A7,Y   Puffernummer in Tabelle
D21A   C6 6F      DEC $6F       Pufferanzahl
D21C   30 08      BMI $D226     schon fertig ?
D21E   20 8E D2   JSR $D28E     Puffer suchen
D221   30 EC      BMI $D20F     nicht gefunden ?
D223   99 AE 00   STA $00AE,Y   Puffernummer in Tabelle
D226   60         RTS

****************************   Kanal schließen
D227   A5 83      LDA $83       Sekundäradresse
D229   C9 0F      CMP #$0F      15 ?
D22B   D0 01      BNE $D22E     nein
D22D   60         RTS           sonst schon fertig

D22E   A6 83      LDX $83
D230   BD 2B 02   LDA $022B,X   Kanalnummer
D233   C9 FF      CMP #$FF      nicht zugeordnet ?
D235   F0 22      BEQ $D259     dann fertig
D237   29 3F      AND #$3F
D239   85 82      STA $82       Kanalnummer
D23B   A9 FF      LDA #$FF
D23D   9D 2B 02   STA $022B,X   Zuodrnung in Tabelle löschen
D240   A6 82      LDX $82
D242   A9 00      LDA #$00
D244   95 F2      STA $F2,X     READ und WRITE-Flag löschen
D246   20 5A D2   JSR $D25A     Puffer freigeben
D249   A6 82      LDX $82       Kanalnummer
D24B   A9 01      LDA #$01      Bit 0 setzen
D24D   CA         DEX
D24E   30 03      BMI $D253     auf richtige Position schieben
D250   0A         ASL  A
D251   D0 FA      BNE $D24D
D253   0D 56 02   ORA $0256     und im Belegungsregister freigeben
D256   8D 56 02   STA $0256
```

```
D259   60          RTS

****************************** Puffer freigeben
D25A   A6 82       LDX $82      Kanalnummer
D25C   B5 A7       LDA $A7,X    Puffernummer
D25E   C9 FF       CMP #$FF
D260   F0 09       BEQ $D26B    nicht zugeordnet ?
D262   48          PHA
D263   A9 FF       LDA #$FF
D265   95 A7       STA $A7,X    Pufferzuordnung löschen
D267   68          PLA
D268   20 F3 D2    JSR $D2F3    Puffer im Belegungsregister löschen
D26B   A6 82       LDX $82      Kanalnummer
D26D   B5 AE       LDA $AE,X
D26F   C9 FF       CMP #$FF     in zweiter Tabelle zugeordnet ?
D271   F0 09       BEQ $D27C    nein
D273   48          PHA
D274   A9 FF       LDA #$FF
D276   95 AE       STA $AE,X    Zuordung löschen
D278   68          PLA
D279   20 F3 D2    JSR $D2F3    Puffer im Belegungsregister löschen
D27C   A6 82       LDX $82      Kanalnummer
D27E   B5 CD       LDA $CD,X
D280   C9 FF       CMP #$FF     in dritter Tabelle zugeordnet ?
D282   F0 09       BEQ $D28D    nein
D284   48          PHA
D285   A9 FF       LDA #$FF
D287   95 CD       STA $CD,X    Zuordung löschen
D289   68          PLA
D28A   20 F3 D2    JSR $D2F3    Puffer im Belegungsregister löschen
D28D   60          RTS

****************************** Puffer suchen
D28E   98          TYA
D28F   48          PHA
D290   A0 01       LDY #$01
D292   20 BA D2    JSR $D2BA
D295   10 0C       BPL $D2A3
D297   88          DEY
D298   20 BA D2    JSR $D2BA
D29B   10 06       BPL $D2A3
D29D   20 39 D3    JSR $D339
D2A0   AA          TAX
D2A1   30 13       BMI $D2B6
D2A3   B5 00       LDA $00,X
D2A5   30 FC       BMI $D2A3
D2A7   A5 7F       LDA $7F
D2A9   95 00       STA $00,X
D2AB   9D 5B 02    STA $025B,X
D2AE   8A          TXA
D2AF   0A          ASL  A
D2B0   A8          TAY
D2B1   A9 02       LDA #$02
D2B3   99 99 00    STA $0099,Y
D2B6   68          PLA
D2B7   A8          TAY
```

```
D2B8   8A          TXA
D2B9   60          RTS

D2BA   A2 07       LDX #$07
D2BC   B9 4F 02    LDA $024F,Y
D2BF   3D E9 EF    AND $EFE9,X    Bit löschen
D2C2   F0 04       BEQ $D2C8
D2C4   CA          DEX
D2C5   10 F5       BPL $D2BC
D2C7   60          RTS

D2C8   B9 4F 02    LDA $024F,Y
D2CB   5D E9 EF    EOR $EFE9,X    Bit umdrehen
D2CE   99 4F 02    STA $024F,Y
D2D1   8A          TXA            Puffernummer
D2D2   88          DEY
D2D3   30 03       BMI $D2D8
D2D5   18          CLC
D2D6   69 08       ADC #$08
D2D8   AA          TAX            Puffernummer
D2D9   60          RTS
D2DA   A6 82       LDX $82
D2DC   B5 A7       LDA $A7,X
D2DE   30 09       BMI $D2E9
D2E0   8A          TXA
D2E1   18          CLC
D2E2   69 07       ADC #$07
D2E4   AA          TAX
D2E5   B5 A7       LDA $A7,X
D2E7   10 F0       BPL $D2D9
D2E9   C9 FF       CMP #$FF
D2EB   F0 EC       BEQ $D2D9
D2ED   48          PHA
D2EE   A9 FF       LDA #$FF
D2F0   95 A7       STA $A7,X
D2F2   68          PLA
D2F3   29 0F       AND #$0F
D2F5   A8          TAY            Puffernummer
D2F6   C8          INY
D2F7   A2 10       LDX #$10       16
D2F9   6E 50 02    ROR $0250
D2FC   6E 4F 02    ROR $024F      16-Bit Belegungsregister rotieren
D2FF   88          DEY
D300   D0 01       BNE $D303
D302   18          CLC            Bit für Puffer löschen
D303   CA          DEX
D304   10 F3       BPL $D2F9
D306   60          RTS

****************************      alle Kanäle schließen
D307   A9 0E       LDA #$0E       14
D309   85 83       STA $83        Sekundäradresse
D30B   20 27 D2    JSR $D227      Kanal schließen
D30E   C6 83       DEC $83        nächste Sekundäradresse
D310   D0 F9       BNE $D30B
D312   60          RTS
```

```
*****************************  alle Kanäle des anderen Drives schließen
D313   A9 0E      LDA #$0E      14
D315   85 83      STA $83       Sekundäradresse
D317   A6 83      LDX $83
D319   BD 2B 02   LDA $022B,X   Zuordnungtabelle
D31C   C9 FF      CMP #$FF      Kanal zugeordnet ?
D31E   F0 14      BEQ $D334     nein
D320   29 3F      AND #$3F
D322   85 82      STA $82       Kanalnummer
D324   20 93 DF   JSR $DF93     Puffernummer holen
D327   AA         TAX
D328   BD 5B 02   LDA $025B,X   Drivenummer
D32B   29 01      AND #$01      isolieren
D32D   C5 7F      CMP $7F       gleich aktuelle Drivenummer ?
D32F   D0 03      BNE $D334     nein
D331   20 27 D2   JSR $D227     Kanal schließen
D334   C6 83      DEC $83       nächsten Kanal
D336   10 DF      BPL $D317
D338   60         RTS

*****************************
D339   A5 6F      LDA $6F
D33B   48         PHA
D33C   A0 00      LDY #$00
D33E   B6 FA      LDX $FA,Y
D340   B5 A7      LDA $A7,X
D342   10 04      BPL $D348
D344   C9 FF      CMP #$FF
D346   D0 16      BNE $D35E
D348   8A         TXA
D349   18         CLC
D34A   69 07      ADC #$07
D34C   AA         TAX
D34D   B5 A7      LDA $A7,X
D34F   10 04      BPL $D355
D351   C9 FF      CMP #$FF
D353   D0 09      BNE $D35E
D355   C8         INY
D356   C0 05      CPY #$05
D358   90 E4      BCC $D33E
D35A   A2 FF      LDX #$FF
D35C   D0 1C      BNE $D37A
D35E   86 6F      STX $6F
D360   29 3F      AND #$3F
D362   AA         TAX
D363   B5 00      LDA $00,X
D365   30 FC      BMI $D363
D367   C9 02      CMP #$02
D369   90 08      BCC $D373
D36B   A6 6F      LDX $6F
D36D   E0 07      CPX #$07
D36F   90 D7      BCC $D348
D371   B0 E2      BCS $D355

D373   A4 6F      LDY $6F
D375   A9 FF      LDA #$FF
```

```
D377   99 A7 00   STA $00A7,Y
D37A   68         PLA
D37B   85 6F      STA $6F
D37D   8A         TXA
D37E   60         RTS

****************************  Kanal suchen und belegen
D37F   A0 00      LDY #$00
D381   A9 01      LDA #$01      Bit 0 setzen
D383   2C 56 02   BIT $0256
D386   D0 09      BNE $D391     Kanal frei ?
D388   C8         INY
D389   0A         ASL  A        Bit nach links schieben
D38A   D0 F7      BNE $D383     alle Kanäle geprüft ?
D38C   A9 70      LDA #$70
D38E   4C C8 C1   JMP $C1C8     70,'no channel'

D391   49 FF      EOR #$FF      Bitmuster umdrehen
D393   2D 56 02   AND $0256     Bit löschen
D396   8D 56 02   STA $0256     Kanal belegen
D399   98         TYA
D39A   60         RTS

****************************  Byte zur Ausgabe holen
D39B   20 EB D0   JSR $D0EB     Kanal zum Lesen öffnen
D39E   20 00 C1   JSR $C100     LED einschalten
D3A1   20 AA D3   JSR $D3AA     Byte ins Ausgaberegister holen
D3A4   A6 82      LDX $82       Kanalnummer
D3A6   BD 3E 02   LDA $023E,X   Byte holen
D3A9   60         RTS

D3AA   A6 82      LDX $82       Kanalnummer
D3AC   20 25 D1   JSR $D125     Dateityp prüfen
D3AF   D0 03      BNE $D3B4     keine Rel-Datei ?
D3B1   4C 20 E1   JMP $E120     Byte aus Rel-Datei holen

D3B4   A5 83      LDA $83       Sekundäradresse
D3B6   C9 0F      CMP #$0F      15
D3B8   F0 5A      BEQ $D414     ja, Fehlerkanal lesen
D3BA   B5 F2      LDA $F2,X
D3BC   29 08      AND #$08      Endeflag gesetzt ?
D3BE   D0 13      BNE $D3D3     nein
D3C0   20 25 D1   JSR $D125     Dateityp prüfen
D3C3   C9 07      CMP #$07      Direktzugriffsdatei ?
D3C5   D0 07      BNE $D3CE     nein
D3C7   A9 89      LDA #$89      READ und WRITE-Flag setzen
D3C9   95 F2      STA $F2,X
D3CB   4C DE D3   JMP $D3DE

D3CE   A9 00      LDA #$00
D3D0   95 F2      STA $F2,X     READ und WRITE-Flag löschen
D3D2   60         RTS

D3D3   A5 83      LDA $83       Sekundäradresse
D3D5   F0 32      BEQ $D409     Null, LOAD ?
D3D7   20 25 D1   JSR $D125     Dateityp prüfen
```

```
D3DA   C9 04      CMP #$04      Rel-Datei oder Direktzugriff ?
D3DC   90 22      BCC $D400     nein
D3DE   20 2F D1   JSR $D12F     Puffer- und Kanalnummer holen
D3E1   B5 99      LDA $99,X     Pufferzeiger
D3E3   D9 44 02   CMP $0244,Y   gleich Endezeiger ?
D3E6   D0 04      BNE $D3EC     nein
D3E8   A9 00      LDA #$00
D3EA   95 99      STA $99,X     Pufferzeiger auf null
D3EC   F6 99      INC $99,X     Pufferzeiger erhöhen
D3EE   A1 99      LDA ($99,X)   Byte aus Puffer holen
D3F0   99 3E 02   STA $023E,Y   ins Ausgaberegister
D3F3   B5 99      LDA $99,X     Pufferzeiger
D3F5   D9 44 02   CMP $0244,Y   gleich Endezeiger ?
D3F8   D0 05      BNE $D3FF     nein
D3FA   A9 81      LDA #$81
D3FC   99 F2 00   STA $00F2,Y   Flags setzen
D3FF   60         RTS

D400   20 56 D1   JSR $D156     Byte aus Puffer holen
D403   A6 82      LDX $82       Kanalnummer
D405   9D 3E 02   STA $023E,X   Byte in Ausgaberegister
D408   60         RTS

D409   AD 54 02   LDA $0254     Flag für Directory ?
D40C   F0 F2      BEQ $D400     nein
D40E   20 67 ED   JSR $ED67     Directoryzeile erzeugen
D411   4C 03 D4   JMP $D403

D414   20 E8 D4   JSR $D4E8     Pufferzeiger setzen
D417   C9 D4      CMP #$D4      zeigt er vor Puffer für Fehlermeldung ?
D419   D0 18      BNE $D433     nein
D41B   A5 95      LDA $95
D41D   C9 02      CMP #$02
D41F   D0 12      BNE $D433
D421   A9 0D      LDA #$0D      CR
D423   85 85      STA $85       in Ausgaberegister
D425   20 23 C1   JSR $C123     Fehlerflags löschen
D428   A9 00      LDA #$00
D42A   20 C1 E6   JSR $E6C1     'ok' Meldung erzeugen
D42D   C6 A5      DEC $A5       Pufferzeiger zurücksetzen
D42F   A9 80      LDA #$80      READ-Flag setzen
D431   D0 12      BNE $D445

D433   20 37 D1   JSR $D137     Byte aus Puffer holen
D436   85 85      STA $85       ins Ausgaberegister
D438   D0 09      BNE $D443
D43A   A9 D4      LDA #$D4
D43C   20 C8 D4   JSR $D4C8     Pufferzeiger vor Fehlerpuffer setzen
D43F   A9 02      LDA #$02
D441   95 9A      STA $9A,X     Hi-Adresse
D443   A9 88      LDA #$88      READ-Flag setzen
D445   85 F7      STA $F7
D447   A5 85      LDA $85       Datenbyte
D449   8D 43 02   STA $0243     ins Ausgaberegister
D44C   60         RTS
```

```
******************************  nächsten Block lesen
D44D   20 93 DF   JSR $DF93     Puffernummer holen
D450   0A         ASL  A        mal 2
D451   AA         TAX
D452   A9 00      LDA #$00
D454   95 99      STA $99,X     Pufferzeiger auf Null
D456   A1 99      LDA ($99,X)   erstes Byte aus Puffer holen
D458   F0 05      BEQ $D45F     kein Folgeblock ?
D45A   D6 99      DEC $99,X     Pufferzeiger auf -1
D45C   4C 56 D1   JMP $D156     nächsten Block lesen
D45F   60         RTS

******************************  Block lesen
D460   A9 80      LDA #$80      Befehlskode für Lesen
D462   D0 02      BNE $D466

******************************  Block schreiben
D464   A9 90      LDA #$90      Befehlskode für schreiben
D466   05 7F      ORA $7F       Drivenummer
D468   8D 4D 02   STA $024D     Kode merken
D46B   A5 F9      LDA $F9
D46D   20 D3 D6   JSR $D6D3     Parameter an Disk-Controller
D470   A6 F9      LDX $F9
D472   4C 93 D5   JMP $D593     Befehl ausführen

******************************  Puffer belegen und Block lesen
D475   A9 01      LDA #$01
D477   8D 4A 02   STA $024A     Dateityp auf sequentiell
D47A   A9 11      LDA #$11      17
D47C   85 83      STA $83       Sekundäradresse
D47E   20 46 DC   JSR $DC46     Puffer belegen und Block lesen
D481   A9 02      LDA #$02
D483   4C C8 D4   JMP $D4C8     Pufferzeiger auf 2

******************************  neuen Block anlegen
D486   A9 12      LDA #$12      18
D488   85 83      STA $83       Sekundäradresse
D48A   4C DA DC   JMP $DCDA     neuen Block anlegen

******************************  Directoryblock schreiben
D48D   20 3B DE   JSR $DE3B     Track und Sektornummer holen
D490   A9 01      LDA #$01
D492   85 6F      STA $6F       ein Block
D494   A5 69      LDA $69       Schrittweite 10 bei Blockbelegung merken
D496   48         PHA
D497   A9 03      LDA #$03      durch 3 bei Directory ersetzen
D499   85 69      STA $69
D49B   20 2D F1   JSR $F12D     freien Block in BAM suchen
D49E   68         PLA
D49F   85 69      STA $69       Schrittweite zurückholen
D4A1   A9 00      LDA #$00
D4A3   20 C8 D4   JSR $D4C8     Pufferzeiger auf Null
D4A6   A5 80      LDA $80
D4A8   20 F1 CF   JSR $CFF1     Tracknummer in Puffer
D4AB   A5 81      LDA $81
D4AD   20 F1 CF   JSR $CFF1     Sektornummer in Puffer
```

```
D4B0   20 C7 D0   JSR $D0C7     Block auf Diskette schreiben
D4B3   20 99 D5   JSR $D599     und prüfen
D4B6   A9 00      LDA #$00
D4B8   20 C8 D4   JSR $D4C8     Pufferzeiger auf Null
D4BB   20 F1 CF   JSR $CFF1     Puffer mit Nullen füllen
D4BE   D0 FB      BNE $D4BB
D4C0   20 F1 CF   JSR $CFF1     Null als Folgetrack
D4C3   A9 FF      LDA #$FF
D4C5   4C F1 CF   JMP $CFF1     $FF als Anzahl der Bytes

******************************  Pufferzeiger setzen
D4C8   85 6F      STA $6F       Zeiger merken
D4CA   20 93 DF   JSR $DF93     Puffernummer holen
D4CD   0A         ASL  A        mal 2
D4CE   AA         TAX
D4CF   B5 9A      LDA $9A,X     Pufferzeiger hi
D4D1   85 95      STA $95
D4D3   A5 6F      LDA $6F
D4D5   95 99      STA $99,X     Pufferzeiger lo, neuer Wert
D4D7   85 94      STA $94
D4D9   60         RTS

******************************  Interne Kanäle schließen
D4DA   A9 11      LDA #$11      17
D4DC   85 83      STA $83
D4DE   20 27 D2   JSR $D227     Kanal schließen
D4E1   A9 12      LDA #$12      18
D4E3   85 83      STA $83
D4E5   4C 27 D2   JMP $D227     Kanal schließen

******************************  Pufferzeiger setzen
D4E8   20 93 DF   JSR $DF93     Puffernummer holen
D4EB   0A         ASL  A
D4EC   AA         TAX
D4ED   B5 9A      LDA $9A,X     Pufferzeiger hi
D4EF   85 95      STA $95
D4F1   B5 99      LDA $99,X     Pufferzeiger lo
D4F3   85 94      STA $94
D4F5   60         RTS

******************************  Byte aus Puffer holen
D4F6   85 71      STA $71       Zeiger lo
D4F8   20 93 DF   JSR $DF93     Puffernummer holen
D4FB   AA         TAX
D4FC   BD E0 FE   LDA $FEE0,X   Hi-Byte Pufferadresse
D4FF   85 72      STA $72       Zeiger hi
D501   A0 00      LDY #$00
D503   B1 71      LDA ($71),Y   Byte aus Puffer holen
D505   60         RTS

******************************  Track und Sektornummer überprüfen
D506   BD 5B 02   LDA $025B,X   Befehlskode für Disk-Controller
D509   29 01      AND #$01      Drivenummer
D50B   0D 4D 02   ORA $024D     plus Befehlskode
D50E   48         PHA           merken
D50F   86 F9      STX $F9       Puffernummer
```

```
D511   8A          TXA
D512   0A          ASL  A         mal 2
D513   AA          TAX
D514   B5 07       LDA $07,X      Sektor
D516   8D 4D 02    STA $024D      merken
D519   B5 06       LDA $06,X      Track
D51B   F0 2D       BEQ $D54A      66, 'illegal track or sector'
D51D   CD D7 FE    CMP $FED7      36, höchste Tracknummer + 1
D520   B0 28       BCS $D54A      66, 'illegal track or sector'
D522   AA          TAX
D523   68          PLA            Befehlskode
D524   48          PHA
D525   29 F0       AND #$F0
D527   C9 90       CMP #$90       Kode für Schreiben ?
D529   D0 4F       BNE $D57A      nein
D52B   68          PLA
D52C   48          PHA
D52D   4A          LSR  A
D52E   B0 05       BCS $D535
D530   AD 01 01    LDA $0101
D533   90 03       BCC $D538
D535   AD 02 01    LDA $0102
D538   F0 05       BEQ $D53F
D53A   CD D5 FE    CMP $FED5      'A', Formatkennzeichen
D53D   D0 33       BNE $D572      73, 'cbm dos v2.6 1541'
D53F   8A          TXA            Tracknummer
D540   20 4B F2    JSR $F24B      maximale Sektornummer holen
D543   CD 4D 02    CMP $024D      mit Sektornummer vergleichen
D546   F0 02       BEQ $D54A      gleich, dann Fehler
D548   B0 30       BCS $D57A      kleiner ?
D54A   20 52 D5    JSR $D552      Track und Sektornummer holen
D54D   A9 66       LDA #$66
D54F   4C 45 E6    JMP $E645      66, 'illegal track or sector'

******************************    Track und Sektornummer holen
D552   A5 F9       LDA $F9        Puffernummer
D554   0A          ASL  A         *2
D555   AA          TAX            als Index
D556   B5 06       LDA $06,X
D558   85 80       STA $80        Track
D55A   B5 07       LDA $07,X
D55C   85 81       STA $81        Sektor
D55E   60          RTS

D55F   A5 80       LDA $80        Track
D561   F0 EA       BEQ $D54D      null, dann Fehler
D563   CD D7 FE    CMP $FED7      36, maximale Tracknummer + 1
D566   B0 E5       BCS $D54D      66, 'illegal track or sector'
D568   20 4B F2    JSR $F24B      maximale Sektornummer holen
D56B   C5 81       CMP $81        Sektor
D56D   F0 DE       BEQ $D54D
D56F   90 DC       BCC $D54D      Fehler
D571   60          RTS

D572   20 52 D5    JSR $D552      Track und Sektornummer holen
D575   A9 73       LDA #$73
```

```
D577   4C 45 E6   JMP $E645      73, 'cbm dos v2.6 1541'

D57A   A6 F9      LDX $F9       Puffernummer
D57C   68         PLA
D57D   8D 4D 02   STA $024D     Befehlskode für Disk-Controller
D580   95 00      STA $00,X     in Befehlsregister
D582   9D 5B 02   STA $025B,X   und in Tabelle schreiben
D585   60         RTS

******************************  Block lesen
D586   A9 80      LDA #$80      Kode für Lesen
D588   D0 02      BNE $D58C

******************************  Block schreiben
D58A   A9 90      LDA #$90      Kode für schreiben
D58C   05 7F      ORA $7F       Drivenummer
D58E   A6 F9      LDX $F9       Puffernummer
D590   8D 4D 02   STA $024D
D593   AD 4D 02   LDA $024D     Befehlskode
D596   20 0E D5   JSR $D50E     Track und Sektor prüfen und an Disk-Controller

******************************  Ausführung prüfen
D599   20 A6 D5   JSR $D5A6     Ausführung prüfen
D59C   B0 FB      BCS $D599     Ende abwarten
D59E   48         PHA           Rückmeldungskode
D59F   A9 00      LDA #$00
D5A1   8D 98 02   STA $0298     Fehlerflag löschen
D5A4   68         PLA
D5A5   60         RTS

D5A6   B5 00      LDA $00,X     Befehlskode (Bit 7) noch im Register ?
D5A8   30 1A      BMI $D5C4     ja
D5AA   C9 02      CMP #$02      Rückmeldung kleiner 2
D5AC   90 14      BCC $D5C2     dann fehlerfreie Durchführung
D5AE   C9 08      CMP #$08      8
D5B0   F0 08      BEQ $D5BA     dann Write Protect
D5B2   C9 0B      CMP #$0B      11
D5B4   F0 04      BEQ $D5BA     dann ID mismatch
D5B6   C9 0F      CMP #$0F      15
D5B8   D0 0C      BNE $D5C6
D5BA   2C 98 02   BIT $0298
D5BD   30 03      BMI $D5C2
D5BF   4C 3F D6   JMP $D63F     Fehlermeldung erzeugen
D5C2   18         CLC           Ausführung beendet
D5C3   60         RTS

D5C4   38         SEC           Ausführung noch nicht beendet
D5C5   60         RTS

D5C6   98         TYA
D5C7   48         PHA
D5C8   A5 7F      LDA $7F       Drivenummer
D5CA   48         PHA
D5CB   BD 5B 02   LDA $025B,X
D5CE   29 01      AND #$01      Drivenummer
D5D0   85 7F      STA $7F
```

```
D5D2   A8          TAY
D5D3   B9 CA FE    LDA $FECA,Y   Bitmuster für Drive
D5D6   8D 6D 02    STA $026D
D5D9   20 A6 D6    JSR $D6A6     Leseversuch
D5DC   C9 02       CMP #$02      Rückmeldung
D5DE   B0 03       BCS $D5E3     nicht ok ?
D5E0   4C 6D D6    JMP $D66D     fertig
D5E3   BD 5B 02    LDA $025B,X   Befehlskode
D5E6   29 F0       AND #$F0      isolieren
D5E8   48          PHA
D5E9   C9 90       CMP #$90      Kode für Schreiben
D5EB   D0 07       BNE $D5F4     nein
D5ED   A5 7F       LDA $7F       Drivenummer
D5EF   09 B8       ORA #$B8
D5F1   9D 5B 02    STA $025B,X
D5F4   24 6A       BIT $6A
D5F6   70 39       BVS $D631
D5F8   A9 00       LDA #$00
D5FA   8D 99 02    STA $0299     Zähler für Suche neben dem Track
D5FD   8D 9A 02    STA $029A
D600   AC 99 02    LDY $0299     Zähler
D603   AD 9A 02    LDA $029A
D606   38          SEC
D607   F9 DB FE    SBC $FEDB,Y   Konstanten für Leseversuche neben dem Track
D60A   8D 9A 02    STA $029A
D60D   B9 DB FE    LDA $FEDB,Y
D610   20 76 D6    JSR $D676     Kopf neben dem Track positionieren
D613   EE 99 02    INC $0299     Zähler erhöhen
D616   20 A6 D6    JSR $D6A6     Leseversuch
D619   C9 02       CMP #$02      Rückmeldung
D61B   90 08       BCC $D625     kleiner 2, ok ?
D61D   AC 99 02    LDY $0299     Zähler laden
D620   B9 DB FE    LDA $FEDB,Y   Konstanten holen
D623   D0 DB       BNE $D600     noch nicht Null (Tabellenende) ?
D625   AD 9A 02    LDA $029A
D628   20 76 D6    JSR $D676     Kopf positionieren
D62B   B5 00       LDA $00,X
D62D   C9 02       CMP #$02      Rückmeldung
D62F   90 2B       BCC $D65C     ok ?
C631   24 6A       BIT $6A
D633   10 0F       BPL $D644
D635   68          PLA           Befehlskode
D636   C9 90       CMP #$90      für Schreiben ?
D638   D0 05       BNE $D63F     nein
D63A   05 7F       ORA $7F       Drivenummer
D63C   9D 5B 02    STA $025B,X   Befehlskode in Tabelle
D63F   B5 00       LDA $00,X     Rückmeldung
D641   20 0A E6    JSR $E60A     Fehlermeldung setzen
D644   68          PLA
D645   2C 98 02    BIT $0298
D648   30 23       BMI $D66D
D64A   48          PHA
D64B   A9 C0       LDA #$C0      Befehlskode für Kopfpositionierung
D64D   05 7F       ORA $7F       Drivenummer
D64F   95 00       STA $00,X     in Befehlsregister
D651   B5 00       LDA $00,X
```

```
D653  30 FC      BMI $D651      Ausführung abwarten
D655  20 A6 D6   JSR $D6A6      Befehlsausführung nochmal versuchen
D658  C9 02      CMP #$02       Rückmeldung
D65A  B0 D9      BCS $D635      fehlerhaft ?
D65C  68         PLA
D65D  C9 90      CMP #$90       Befehlskode für Schreiben
D65F  D0 0C      BNE $D66D      nein
D661  05 7F      ORA $7F        Drivenummer
D663  9D 5B 02   STA $025B,X    in Tabelle
D666  20 A6 D6   JSR $D6A6      Ausführung nochmal versuchen
D669  C9 02      CMP #$02       Rückmeldung
D66B  B0 D2      BCS $D63F      Fehler ?
D66D  68         PLA
D66E  85 7F      STA $7F        Drivenummer zurückholen
D670  68         PLA
D671  A8         TAY
D672  B5 00      LDA $00,X      Fehlerkode
D674  18         CLC            Flag für Ausführung beendet
D675  60         RTS

D676  C9 00      CMP #$00
D678  F0 18      BEQ $D692
D67A  30 0C      BMI $D688
D67C  A0 01      LDY #$01
D67E  20 93 D6   JSR $D693      Daten für Kopfpositionierung übergeben
D681  38         SEC
D682  E9 01      SBC #$01
D684  D0 F6      BNE $D67C
D686  F0 0A      BEQ $D692

D688  A0 FF      LDY #$FF
D68A  20 93 D6   JSR $D693      Daten für Kopfpositionierung übergeben
D68D  18         CLC
D68E  69 01      ADC #$01
D690  D0 F6      BNE $D688
D692  60         RTS

D693  48         PHA
D694  98         TYA
D695  A4 7F      LDY $7F        Drivenummer
D697  99 FE 02   STA $02FE,Y
D69A  D9 FE 02   CMP $02FE,Y    Rückmeldung des Disk-Controllers abwarten
D69D  F0 FB      BEQ $D69A
D69F  A9 00      LDA #$00
D6A1  99 FE 02   STA $02FE,Y
D6A4  68         PLA
D6A5  60         RTS

D6A6  A5 6A      LDA $6A        Maximalzahl der Wiederholungen
D6A8  29 3F      AND #$3F
D6AA  A8         TAY
D6AB  AD 6D 02   LDA $026D      Bit für LED
D6AE  4D 00 1C   EOR $1C00
D6B1  8D 00 1C   STA $1C00      LED umschalten
D6B4  BD 5B 02   LDA $025B,X    Befehl
D6B7  95 00      STA $00,X      an Disk-Controller übergeben
```

```
D6B9   B5 00      LDA $00,X     und Rückmeldung
D6BB   30 FC      BMI $D6B9     abwarten
D6BD   C9 02      CMP #$02      ok ?
D6BF   90 03      BCC $D6C4     ja
D6C1   88         DEY           Zähler erniedrigen
D6C2   D0 E7      BNE $D6AB     nochmal versuchen
D6C4   48         PHA
D6C5   AD 6D 02   LDA $026D
D6C8   0D 00 1C   ORA $1C00     LED aus
D6CB   8D 00 1C   STA $1C00
D6CE   68         PLA
D6CF   60         RTS

****************************  Parameter an Disk-Controller übergeben
D6D0   20 93 DF   JSR $DF93     Puffernummer holen
D6D3   0A         ASL  A
D6D4   A8         TAY
D6D5   A5 80      LDA $80       Tracknummer
D6D7   99 06 00   STA $0006,Y   übergeben
D6DA   A5 81      LDA $81       Sektornummer
D6DC   99 07 00   STA $0007,Y   übergeben
D6DF   A5 7F      LDA $7F       Drivenummer
D6E1   0A         ASL  A        mal 2
D6E2   AA         TAX           nach X
D6E3   60         RTS

****************************  Datei in Directory eintragen
D6E4   A5 83      LDA $83       Sekundäradresse
D6E6   48         PHA
D6E7   A5 82      LDA $82       Kanalnummer
D6E9   48         PHA
D6EA   A5 81      LDA $81       Sektornummer
D6EC   48         PHA
D6ED   A5 80      LDA $80       Tracknummer
D6EF   48         PHA           merken
D6F0   A9 11      LDA #$11
D6F2   85 83      STA $83       Sekundäradresse 17
D6F4   20 3B DE   JSR $DE3B     Track und Sektornummer holen
D6F7   AD 4A 02   LDA $024A     Dateityp
D6FA   48         PHA           merken
D6FB   A5 E2      LDA $E2       Drivenummer
D6FD   29 01      AND #$01
D6FF   85 7F      STA $7F       setzen
D701   A6 F9      LDX $F9       Puffernummer
D703   5D 5B 02   EOR $025B,X
D706   4A         LSR  A
D707   90 0C      BCC $D715     gleiche Drivenummer ?
D709   A2 01      LDX #$01
D70B   8E 92 02   STX $0292     Zeiger in Directory
D70E   20 AC C5   JSR $C5AC     Directory laden und ersten Eintrag suchen
D711   F0 1D      BEQ $D730     nicht gefunden ?
D713   D0 28      BNE $D73D     gefunden ?

D715   AD 91 02   LDA $0291     Sektornummer im Directory
D718   F0 0C      BEQ $D726     gleich null
D71A   C5 81      CMP $81       gleiche Sektornummer ?
```

```
D71C   F0 1F      BEQ $D73D     ja
D71E   85 81      STA $81       Sektornummer merken
D720   20 60 D4   JSR $D460     Block lesen
D723   4C 3D D7   JMP $D73D

D726   A9 01      LDA #$01
D728   8D 92 02   STA $0292     Zeiger auf eins
D72B   20 17 C6   JSR $C617     nächsten Eintrag im Directory suchen
D72E   D0 0D      BNE $D73D     gefunden ?
D730   20 8D D4   JSR $D48D     Directoryblock schreiben
D733   A5 81      LDA $81       Sektornummer
D735   8D 91 02   STA $0291
D738   A9 02      LDA #$02
D73A   8D 92 02   STA $0292     Zeiger auf 2
D73D   AD 92 02   LDA $0292
D740   20 C8 D4   JSR $D4C8     Pufferzeiger setzen
D743   68         PLA
D744   8D 4A 02   STA $024A     Dateityp
D747   C9 04      CMP #$04      Rel-Datei ?
D749   D0 02      BNE $D74D     nein
D74B   09 80      ORA #$80      Bit 7 setzen
D74D   20 F1 CF   JSR $CFF1     und in Puffer schreiben
D750   68         PLA
D751   8D 80 02   STA $0280     Folgetrack
D754   20 F1 CF   JSR $CFF1     in Puffer
D757   68         PLA
D758   8D 85 02   STA $0285     Folgesektor
D75B   20 F1 CF   JSR $CFF1     in Puffer
D75E   20 93 DF   JSR $DF93     Puffernummer holen
D761   A8         TAY
D762   AD 7A 02   LDA $027A     Zeiger auf Drivenummer
D765   AA         TAX
D766   A9 10      LDA #$10      16, Länge des Filenamens
D768   20 6E C6   JSR $C66E     Filenamen in Puffer schreiben
D76B   A0 10      LDY #$10
D76D   A9 00      LDA #$00
D76F   91 94      STA ($94),Y   ab Position 16 mit Nullen füllen
D771   C8         INY
D772   C0 1B      CPY #$1B      schon Position 27 ?
D774   90 F9      BCC $D76F     nein
D776   AD 4A 02   LDA $024A     Dateityp
D779   C9 04      CMP #$04      Rel-Datei
D77B   D0 13      BNE $D790     nein
D77D   A0 10      LDY #$10
D77F   AD 59 02   LDA $0259     Track
D782   91 94      STA ($94),Y
D784   C8         INY
D785   AD 5A 02   LDA $025A     und Sektor
D788   91 94      STA ($94),Y   der Side-Sektoren in Directoryeintrag
D78A   C8         INY
D78B   AD 58 02   LDA $0258     Recordlänge
D78E   91 94      STA ($94),Y   in Directory
D790   20 64 D4   JSR $D464     Block schreiben
D793   68         PLA
D794   85 82      STA $82       Kanalnummer
D796   AA         TAX
```

```
D797   68          PLA
D798   85 83       STA $83        Sekundäradresse
D79A   AD 91 02    LDA $0291
D79D   85 D8       STA $D8
D79F   9D 60 02    STA $0260,X
D7A2   AD 92 02    LDA $0292
D7A5   85 DD       STA $DD
D7A7   9D 66 02    STA $0266,X
D7AA   AD 4A 02    LDA $024A      Dateityp
D7AD   85 E7       STA $E7
D7AF   A5 7F       LDA $7F        Drivenummer
D7B1   85 E2       STA $E2
D7B3   60          RTS

****************************   OPEN-Befehl, Sekundäradresse <> 15
D7B4   A5 83       LDA $83        Sekundäradresse
D7B6   8D 4C 02    STA $024C
D7B9   20 B3 C2    JSR $C2B3      Zeilenlänge holen, Flags löschen
D7BC   8E 2A 02    STX $022A
D7BF   AE 00 02    LDX $0200      erstes Zeichen aus Puffer
D7C2   AD 4C 02    LDA $024C      Sekundäradresse
D7C5   D0 2C       BNE $D7F3      ungleich 0 (LOAD) ?
D7C7   E0 2A       CPX #$2A       '*'
D7C9   D0 28       BNE $D7F3
D7CB   A5 7E       LDA $7E        letzte Tracknummer
D7CD   F0 4D       BEQ $D81C
D7CF   85 80       STA $80        Tracknummer
D7D1   AD 6E 02    LDA $026E      letzte Drivenummer
D7D4   85 7F       STA $7F        Drivenummer
D7D6   85 E2       STA $E2
D7D8   A9 02       LDA #$02
D7DA   85 E7       STA $E7        Dateityp auf Programm
D7DC   AD 6F 02    LDA $026F      letzte Sektornummer
D7DF   85 81       STA $81        Sektor
D7E1   20 00 C1    JSR $C100      LED einschalten
D7E4   20 46 DC    JSR $DC46      Puffer belegen, Block lesen
D7E7   A9 04       LDA #$04       Dateityp
D7E9   05 7F       ORA $7F        Drivenummer
D7EB   A6 82       LDX $82        Kanalnummer
D7ED   99 EC 00    STA $00EC,Y    Flag setzen
D7F0   4C 94 C1    JMP $C194      fertig

D7F3   E0 24       CPX #$24       '$'
D7F5   D0 1E       BNE $D815      nein
D7F7   AD 4C 02    LDA $024C      Sekundäradresse
D7FA   D0 03       BNE $D7FF      ungleich null ?
D7FC   4C 55 DA    JMP $DA55      OPEN $

D7FF   20 D1 C1    JSR $C1D1      Zeile bis zu Ende analysieren
D802   AD 85 FE    LDA $FE85      18, Directorytrack
D805   85 80       STA $80        Track
D807   A9 00       LDA #$00
D809   85 81       STA $81        Sektor 0
D80B   20 46 DC    JSR $DC46      Puffer belegen, Block lesen
D80E   A5 7F       LDA $7F        Drivenummer
D810   09 02       ORA #$02
```

```
D812   4C EB D7   JMP $D7EB     weiter wie oben

D815   E0 23      CPX #$23      '#'
D817   D0 12      BNE $D82B
D819   4C 84 CB   JMP $CB84     Direktzugriffsdatei öffnen

D81C   A9 02      LDA #$02
D81E   8D 96 02   STA $0296     Dateityp Programm
D821   A9 00      LDA #$00
D823   85 7F      STA $7F       Drive 0
D825   8D 8E 02   STA $028E
D828   20 42 D0   JSR $D042     BAM laden
D82B   20 E5 C1   JSR $C1E5     Zeile analysieren
D82E   D0 04      BNE $D834     Doppelpunkt gefunden ?
D830   A2 00      LDX #$00
D832   F0 0C      BEQ $D840
D834   8A         TXA           Komma gefunden ?
D835   F0 05      BEQ $D83C     nein
D837   A9 30      LDA #$30
D839   4C C8 C1   JMP $C1C8     30, 'syntax error'

D83C   88         DEY
D83D   F0 01      BEQ $D840
D83F   88         DEY
D840   8C 7A 02   STY $027A     Zeiger auf Drivenummer
D843   A9 8D      LDA #$8D      Shift CR
D845   20 68 C2   JSR $C268     Zeile bis Ende untersuchen
D848   E8         INX
D849   8E 78 02   STX $0278     Kommazähler
D84C   20 12 C3   JSR $C312     Drivenummer holen
D84F   20 CA C3   JSR $C3CA     Drivenummer prüfen
D852   20 9D C4   JSR $C49D     Dateieintrag im Directory suchen
D855   A2 00      LDX #$00      Defaultwerte
D857   8E 58 02   STX $0258     Recordlänge
D85A   8E 97 02   STX $0297
D85D   8E 4A 02   STX $024A     Dateityp
D860   E8         INX
D861   EC 77 02   CPX $0277     Komma vor Gleichheitszeichen ?
D864   B0 10      BCS $D876     nein
D866   20 09 DA   JSR $DA09     holt Filetyp und Betriebsart
D869   E8         INX
D86A   EC 77 02   CPX $0277     weiteres Komma ?
D86D   B0 07      BCS $D876     nein
D86F   C0 04      CPY #$04
D871   F0 3E      BEQ $D8B1
D873   20 09 DA   JSR $DA09     holt Filetyp und Betriebsart
D876   AE 4C 02   LDX $024C
D879   86 83      STX $83       Sekundäradresse
D87B   E0 02      CPX #$02      größer gleich 2 ?
D87D   B0 12      BCS $D891     ja
D87F   8E 97 02   STX $0297     0 oder 1 (LOAD oder SAVE)
D882   A9 40      LDA #$40
D884   8D F9 02   STA $02F9
D887   AD 4A 02   LDA $024A     Dateityp
D88A   D0 1B      BNE $D8A7     nicht deleted
D88C   A9 02      LDA #$02      Prg
```

```
D88E   8D 4A 02   STA $024A      als Dateityp
D891   AD 4A 02   LDA $024A
D894   D0 11      BNE $D8A7
D896   A5 E7      LDA $E7
D898   29 07      AND #$07       Dateityp aus Befehlszeile holen
D89A   8D 4A 02   STA $024A
D89D   AD 80 02   LDA $0280      Tracknummer
D8A0   D0 05      BNE $D8A7      ungleich null ?
D8A2   A9 01      LDA #$01
D8A4   8D 4A 02   STA $024A      Dateityp sequentiell
D8A7   AD 97 02   LDA $0297      Betriebsart
D8AA   C9 01      CMP #$01       'W'
D8AC   F0 18      BEQ $D8C6      ja
D8AE   4C 40 D9   JMP $D940

D8B1   BC 7A 02   LDY $027A,X    Zeiger hinter zweites Komma
D8B4   B9 00 02   LDA $0200,Y    Wert holen
D8B7   8D 58 02   STA $0258      Recordlänge
D8BA   AD 80 02   LDA $0280      Tracknummer
D8BD   D0 B7      BNE $D876
D8BF   A9 01      LDA #$01       'W'
D8C1   8D 97 02   STA $0297      als Betriebsart
D8C4   D0 B0      BNE $D876

D8C6   A5 E7      LDA $E7        Dateityp
D8C8   29 80      AND #$80       Jokerflag isolieren
D8CA   AA         TAX
D8CB   D0 14      BNE $D8E1      Joker im Namen
D8CD   A9 20      LDA #$20
D8CF   24 E7      BIT $E7        war Datei geschlossen ?
D8D1   F0 06      BEQ $D8D9      ja
D8D3   20 B6 C8   JSR $C8B6      Byte 0 in Puffer und Block schreiben
D8D6   4C E3 D9   JMP $D9E3      Side-Sektor anlegen, fertig

D8D9   AD 80 02   LDA $0280      Tracknummer des ersten Blocks
D8DC   D0 03      BNE $D8E1      schon vorhanden
D8DE   4C E3 D9   JMP $D9E3      Side-Sektor Block anlegen
D8E1   AD 00 02   LDA $0200      erstes Zeichen aus Eingabepuffer
D8E4   C9 40      CMP #$40       '@' Klammeraffe ?
D8E6   F0 0D      BEQ $D8F5      ja
D8E8   8A         TXA
D8E9   D0 05      BNE $D8F0      Joker gesetzt ?
D8EB   A9 63      LDA #$63
D8ED   4C C8 C1   JMP $C1C8      63, 'file exists'
D8F0   A9 33      LDA #$33
D8F2   4C C8 C1   JMP $C1C8      33, 'syntax error'

****************************** Öffnen eines Files mit Überschreiben
D8F5   A5 E7      LDA $E7        Filetyp
D8F7   29 07      AND #$07       isolieren
D8F9   CD 4A 02   CMP $024A
D8FC   D0 67      BNE $D965      Filetypen unterschiedlich ?
D8FE   C9 04      CMP #$04       Rel-File ?
D900   F0 63      BEQ $D965      64, 'file type mismatch'
D902   20 DA DC   JSR $DCDA      neuen Sektor anlegen
D905   A5 82      LDA $82
```

```
D907   8D 70 02   STA $0270     Kanalnummer merken
D90A   A9 11      LDA #$11
D90C   85 83      STA $83       Kanal 17
D90E   20 EB D0   JSR $D0EB     Lesekanal eröffnen
D911   AD 94 02   LDA $0294
D914   20 C8 D4   JSR $D4C8     Pufferzeiger für Directory setzen
D917   A0 00      LDY #$00
D919   B1 94      LDA ($94),Y   Filetyp
D91B   09 20      ORA #$20      Bit 5 setzen, Datei offen
D91D   91 94      STA ($94),Y
D91F   A0 1A      LDY #$1A
D921   A5 80      LDA $80       Track
D923   91 94      STA ($94),Y
D925   C8         INY
D926   A5 81      LDA $81       und Sektor
D928   91 94      STA ($94),Y   beim öffnen mit 'Klammeraffe'
D92A   AE 70 02   LDX $0270     Kanalnummer
D92D   A5 D8      LDA $D8
D92F   9D 60 02   STA $0260,X   Zeiger in Directoryblock
D932   A5 DD      LDA $DD
D934   9D 66 02   STA $0266,X
D937   20 3B DE   JSR $DE3B     Track und Sektornummer holen
D93A   20 64 D4   JSR $D464     Block schreiben
D93D   4C EF D9   JMP $D9EF     Track-, Sektor- und Drivenumme bereitstellen

D940   AD 80 02   LDA $0280     erste Tracknummer
D943   D0 05      BNE $D94A     Datei nicht gelöscht ?
D945   A9 62      LDA #$62
D947   4C C8 C1   JMP $C1C8     62, 'file not found'
D94A   AD 97 02   LDA $0297     Betriebsart
D94D   C9 03      CMP #$03      'M'
D94F   F0 0B      BEQ $D95C     ja, dann kein Test auf nicht geschlossene Datei
D951   A9 20      LDA #$20      Bit 5
D953   24 E7      BIT $E7       in Dateityp testen
D955   F0 05      BEQ $D95C     nicht gesetzt, ok
D957   A9 60      LDA #$60
D959   4C C8 C1   JMP $C1C8     60, 'write file open'
D95C   A5 E7      LDA $E7
D95E   29 07      AND #$07      Dateityp isolieren
D960   CD 4A 02   CMP $024A     Übereinstimmung mit Typ aus Befehl ?
D963   F0 05      BEQ $D96A     ja
D965   A9 64      LDA #$64
D967   4C C8 C1   JMP $C1C8     64, 'file type mismatch'
D96A   A0 00      LDY #$00
D96C   8C 79 02   STY $0279
D96F   AE 97 02   LDX $0297     Betriebsart
D972   E0 02      CPX #$02      'A', Append
D974   D0 1A      BNE $D990     nein
D976   C9 04      CMP #$04      Rel-Datei ?
D978   F0 EB      BEQ $D965     ja, dann Fehler
D97A   B1 94      LDA ($94),Y
D97C   29 4F      AND #$4F      Bit 4,5 und 7 löschen,
D97E   91 94      STA ($94),Y   als offen markieren
D980   A5 83      LDA $83       Kanalnummer merken
D982   48         PHA
D983   A9 11      LDA #$11
```

```
D985   85 83      STA $83        Kanal 17
D987   20 3B DE   JSR $DE3B      Track- Sektornummer holen
D98A   20 64 D4   JSR $D464      Block schreiben
D98D   68         PLA
D98E   85 83      STA $83        Kanalnummer zurückholen
D990   20 A0 D9   JSR $D9A0      Side-Sektor-Parameter übernehmen
D993   AD 97 02   LDA $0297      Betriebsart
D996   C9 02      CMP #$02       'A' Append
D998   D0 55      BNE $D9EF      nein
D99A   20 2A DA   JSR $DA2A      Append vorbereiten
D99D   4C 94 C1   JMP $C194      fertig, Diskstatus bereitstellen

D9A0   A0 13      LDY #$13
D9A2   B1 94      LDA ($94),Y    Track
D9A4   8D 59 02   STA $0259
D9A7   C8         INY
D9A8   B1 94      LDA ($94),Y    und Sektor des ersten Side Sector Blocks
D9AA   8D 5A 02   STA $025A
D9AD   C8         INY
D9AE   B1 94      LDA ($94),Y    Recordlänge
D9B0   AE 58 02   LDX $0258      letzte Recordlänge
D9B3   8D 58 02   STA $0258
D9B6   8A         TXA
D9B7   F0 0A      BEQ $D9C3      letzte Recordlänge null
D9B9   CD 58 02   CMP $0258      Recordlänge gleich ?
D9BC   F0 05      BEQ $D9C3      ja
D9BE   A9 50      LDA #$50
D9C0   20 C8 C1   JSR $C1C8      50, 'record not present'
D9C3   AE 79 02   LDX $0279
D9C6   BD 80 02   LDA $0280,X
D9C9   85 80      STA $80        Track
D9CB   BD 85 02   LDA $0285,X
D9CE   85 81      STA $81        Sektor
D9D0   20 46 DC   JSR $DC46
D9D3   A4 82      LDY $82
D9D5   AE 79 02   LDX $0279
D9D8   B5 D8      LDA $D8,X
D9DA   99 60 02   STA $0260,Y
D9DD   B5 DD      LDA $DD,X
D9DF   99 66 02   STA $0266,Y
D9E2   60         RTS

D9E3   A5 E2      LDA $E2        Drivenummer
D9E5   29 01      AND #$01       isolieren
D9E7   85 7F      STA $7F        Drivenummer
D9E9   20 DA DC   JSR $DCDA      Block anlegen
D9EC   20 E4 D6   JSR $D6E4      Datei im Directory eintragen
D9EF   A5 83      LDA $83        Kanalnummer
D9F1   C9 02      CMP #$02
D9F3   B0 11      BCS $DA06      größer gleich 2 ?
D9F5   20 3E DE   JSR $DE3E      Track und Sektornummer holen
D9F8   A5 80      LDA $80
D9FA   85 7E      STA $7E
D9FC   A5 7F      LDA $7F        Drivenummer
D9FE   8D 6E 02   STA $026E
DA01   A5 81      LDA $81        Sektor
```

```
DA03   8D 6F 02   STA $026F
DA06   4C 99 C1   JMP $C199

****************************  Filetyp und Betriebsarten prüfen
DA09   BC 7A 02   LDY $027A,X   Zeiger in Befehlszeile
DA0C   B9 00 02   LDA $0200,Y   Zeichen aus Zeile holen
DA0F   A0 04      LDY #$04
DA11   88         DEY
DA12   30 08      BMI $DA1C
DA14   D9 B2 FE   CMP $FEB2,Y   Betriebsarten 'R', 'W', 'A', 'M'
DA17   D0 F8      BNE $DA11
DA19   8C 97 02   STY $0297     merken
DA1C   A0 05      LDY #$05
DA1E   88         DEY
DA1F   30 08      BMI $DA29
DA21   D9 B6 FE   CMP $FEB6,Y   Filetypen 'D', 'S', 'P', 'U', 'L'
DA24   D0 F8      BNE $DA1E
DA26   8C 4A 02   STY $024A     merken
DA29   60         RTS

****************************  Vorbereitung für Append
DA2A   20 39 CA   JSR $CA39     Kanal zum Lesen öffnen, Byte holen
DA2D   A9 80      LDA #$80
DA2F   20 A6 DD   JSR $DDA6     letztes Byte ?
DA32   F0 F6      BEQ $DA2A     nein
DA34   20 95 DE   JSR $DE95     Track und Sektornummer holen
DA37   A6 81      LDX $81       Sektornummer
DA39   E8         INX
DA3A   8A         TXA
DA3B   D0 05      BNE $DA42     nicht $FF ?
DA3D   20 A3 D1   JSR $D1A3     Puffer schließen, Block schreiben
DA40   A9 02      LDA #$02
DA42   20 C8 D4   JSR $D4C8     Pufferzeiger auf 2
DA45   A6 82      LDX $82       Kanalnummer
DA47   A9 01      LDA #$01
DA49   95 F2      STA $F2,X     Flag für WRITE setzen
DA4B   A9 80      LDA #$80
DA4D   05 82      ORA $82
DA4F   A6 83      LDX $83
DA51   9D 2B 02   STA $022B,X   Kanalnummer in Tabelle
DA54   60         RTS

****************************  OPEN "$"
DA55   A9 0C      LDA #$0C      Befehlsnummer 12
DA57   8D 2A 02   STA $022A
DA5A   A9 00      LDA #$00
DA5C   AE 74 02   LDX $0274
DA5F   CA         DEX
DA60   F0 0B      BEQ $DA6D
DA62   CA         DEX
DA63   D0 21      BNE $DA86
DA65   AD 01 02   LDA $0201     zweites Zeichen
DA68   20 BD C3   JSR $C3BD     Drivenummer holen
DA6B   30 19      BMI $DA86     keine eindeutige Nummer ?
DA6D   85 E2      STA $E2
DA6F   EE 77 02   INC $0277
```

```
DA72   EE 78 02   INC $0278
DA75   EE 7A 02   INC $027A
DA78   A9 80      LDA #$80
DA7A   85 E7      STA $E7       Jokerflag setzen
DA7C   A9 2A      LDA #$2A      '*'
DA7E   8D 00 02   STA $0200     als Dateiname in Befehlspuffer
DA81   8D 01 02   STA $0201
DA84   D0 18      BNE $DA9E     unbedingter Sprung
DA86   20 E5 C1   JSR $C1E5     Eingabezeile bis zum ':' testen
DA89   D0 05      BNE $DA90     gefunden ?
DA8B   20 DC C2   JSR $C2DC     Flags löschen
DA8E   A0 03      LDY #$03
DA90   88         DEY
DA91   88         DEY
DA92   8C 7A 02   STY $027A     Zeiger auf Drivenummer im Befehl
DA95   20 00 C2   JSR $C200     Zeile analysieren
DA98   20 98 C3   JSR $C398     Typ der Datei emitteln
DA9B   20 20 C3   JSR $C320     Drivenummer holen
DA9E   20 CA C3   JSR $C3CA     Drive bei Bedarf initialisieren
DAA1   20 B7 C7   JSR $C7B7     Disketten-Titel bereitstellen
DAA4   20 9D C4   JSR $C49D     Directory laden
DAA7   20 9E EC   JSR $EC9E     Directory erzeugen und bereitstellen
DAAA   20 37 D1   JSR $D137     Byte aus Puffer holen
DAAD   A6 82      LDX $82       Kanalnummer
DAAF   9D 3E 02   STA $023E,X   Byte in Ausgaberegister
DAB2   A5 7F      LDA $7F       Drivenummer
DAB4   8D 8E 02   STA $028E     als letzte Drievnummer merken
DAB7   09 04      ORA #$04
DAB9   95 EC      STA $EC,X     PRG-Flag
DABB   A9 00      LDA #$00
DABD   85 A3      STA $A3       Zeiger in Eingabepuffer rücksetzen
DABF   60         RTS

****************************  CLOSE-Routine
DAC0   A9 00      LDA #$00
DAC2   8D F9 02   STA $02F9
DAC5   A5 83      LDA $83       Sekundäradresse
DAC7   D0 0B      BNE $DAD4     ungleich null ?
DAC9   A9 00      LDA #$00      Sekundäradresse 0, LOAD
DACB   8D 54 02   STA $0254
DACE   20 27 D2   JSR $D227     Kanal schließen
DAD1   4C DA D4   JMP $D4DA     interne Kanäle 17 und 18 schließen

DAD4   C9 0F      CMP #$0F      15
DAD6   F0 14      BEQ $DAEC     ja, alle Kanäle schließen
DAD8   20 02 DB   JSR $DB02     Datei schließen
DADB   A5 83      LDA $83       Sekundäradresse
DADD   C9 02      CMP #$02
DADF   90 F0      BCC $DAD1     kleiner 2 ?
DAE1   AD 6C 02   LDA $026C
DAE4   D0 03      BNE $DAE9
DAE6   4C 94 C1   JMP $C194     Abschluß

DAE9   4C AD C1   JMP $C1AD
DAEC   A9 0E      LDA #$0E      14
DAEE   85 83      STA $83       Sekundäradresse
```

```
DAF0   20 02 DB   JSR $DB02      Datei schließen
DAF3   C6 83      DEC $83        nächste Sekundäradresse
DAF5   10 F9      BPL $DAF0
DAF7   AD 6C 02   LDA $026C
DAFA   D0 03      BNE $DAFF
DAFC   4C 94 C1   JMP $C194      Abschluß
DAFF   4C AD C1   JMP $C1AD

******************************  Datei schließen
DB02   A6 83      LDX $83        Sekundäradresse
DB04   BD 2B 02   LDA $022B,X    Kanalnummer holen
DB07   C9 FF      CMP #$FF       kein Kanal zugeordnet ?
DB09   D0 01      BNE $DB0C
DB0B   60         RTS            nein, dann fertig

DB0C   29 0F      AND #$0F       Kanalnummer isolieren
DB0E   85 82      STA $82
DB10   20 25 D1   JSR $D125      Dateityp prüfen
DB13   C9 07      CMP #$07       Direktzugriff ?
DB15   F0 0F      BEQ $DB26      ja
DB17   C9 04      CMP #$04       Rel-Datei ?
DB19   F0 11      BEQ $DB2C      ja
DB1B   20 07 D1   JSR $D107      Kanal zum Schreiben öffnen
DB1E   B0 09      BCS $DB29      keine Datei zum Schreiben ?
DB20   20 62 DB   JSR $DB62      letzten Block schreiben
DB23   20 A5 DB   JSR $DBA5      Eintrag im Directory und Block schreiben
DB26   20 F4 EE   JSR $EEF4      BAM schreiben
DB29   4C 27 D2   JMP $D227      Kanal schließen

DB2C   20 F1 DD   JSR $DDF1      Puffernummer holen, Block schreiben
DB2F   20 1E CF   JSR $CF1E      Puffer wechseln
DB32   20 CB E1   JSR $E1CB      letzten Side-Sektor holen
DB35   A6 D5      LDX $D5        Side-Sektor-Nummer
DB37   86 73      STX $73
DB39   E6 73      INC $73
DB3B   A9 00      LDA #$00
DB3D   85 70      STA $70
DB3F   85 71      STA $71
DB41   A5 D6      LDA $D6
DB43   38         SEC
DB44   E9 0E      SBC #$0E       minus 14 für Zeiger
DB46   85 72      STA $72
DB48   20 51 DF   JSR $DF51      Blockzahl der Datei berechnen
DB4B   A6 82      LDX $82        Kanalnummer
DB4D   A5 70      LDA $70
DB4F   95 B5      STA $B5,X      Recordnummer lo
DB51   A5 71      LDA $71
DB53   95 BB      STA $BB,X      Recordnummer hi
DB55   A9 40      LDA #$40
DB57   20 A6 DD   JSR $DDA6      Bit 6 gesetzt ?
DB5A   F0 03      BEQ $DB5F      nein
DB5C   20 A5 DB   JSR $DBA5      in Directory eintragen
DB5F   4C 27 D2   JMP $D227      Kanal schließen

******************************  letzten Block schreiben
DB62   A6 82      LDX $82        Kanalnummer
```

```
DB64   B5 B5      LDA $B5,X     Recordnummer lo
DB66   15 BB      ORA $BB,X     Recordnummer hi
DB68   D0 0C      BNE $DB76     ungleich null ?
DB6A   20 E8 D4   JSR $D4E8     Pufferzeiger setzen
DB6D   C9 02      CMP #$02
DB6F   D0 05      BNE $DB76     ungleich 2
DB71   A9 0D      LDA #$0D      CR
DB73   20 F1 CF   JSR $CFF1     in Puffer
DB76   20 E8 D4   JSR $D4E8     Pufferzeiger setzen
DB79   C9 02      CMP #$02      jetzt gleich 2 ?
DB7B   D0 0F      BNE $DB8C     nein
DB7D   20 1E CF   JSR $CF1E     Puffer wechseln
DB80   A6 82      LDX $82       Kanalnummer
DB82   B5 B5      LDA $B5,X     Recordnummer lo
DB84   D0 02      BNE $DB88
DB86   D6 BB      DEC $BB,X     Blockzahl hi
DB88   D6 B5      DEC $B5,X     und Blockzahl lo vermindern
DB8A   A9 00      LDA #$00
DB8C   38         SEC
DB8D   E9 01      SBC #$01      Zeiger auf Ende setzen
DB8F   48         PHA
DB90   A9 00      LDA #$00
DB92   20 C8 D4   JSR $D4C8     Pufferzeiger auf Null
DB95   20 F1 CF   JSR $CFF1     Null in Puffer schreiben
DB98   68         PLA           zweites Byte gleich Zeiger auf Ende
DB99   20 F1 CF   JSR $CFF1     in Puffer schreiben
DB9C   20 C7 D0   JSR $D0C7     Block auf Diskette schreiben
DB9F   20 99 D5   JSR $D599     und überprüfen
DBA2   4C 1E CF   JMP $CF1E     Puffer wechseln

****************************  Directoryeintrag
DBA5   A6 82      LDX $82       Kanalnummer
DBA7   8E 70 02   STX $0270     merken
DBAA   A5 83      LDA $83       Sekundäradresse
DBAC   48         PHA           merken
DBAD   BD 60 02   LDA $0260,X   Sektornummer im Directory
DBB0   85 81      STA $81       setzen
DBB2   BD 66 02   LDA $0266,X   Zeiger in Directory
DBB5   8D 94 02   STA $0294
DBB8   B5 EC      LDA $EC,X
DBBA   29 01      AND #$01
DBBC   85 7F      STA $7F       Drivenummer
DBBE   AD 85 FE   LDA $FE85     18, Directory Track
DBC1   85 80      STA $80       setzen
DBC3   20 93 DF   JSR $DF93     Puffernummer erhöhen
DBC6   48         PHA
DBC7   85 F9      STA $F9
DBC9   20 60 D4   JSR $D460     Directoryblock lesen
DBCC   A0 00      LDY #$00
DBCE   BD E0 FE   LDA $FEE0,X   Pufferadresse
DBD1   85 87      STA $87
DBD3   AD 94 02   LDA $0294     Pufferzeiger
DBD6   85 86      STA $86
DBD8   B1 86      LDA ($86),Y   Dateityp
DBDA   29 20      AND #$20      Datei geschlossen ?
DBDC   F0 43      BEQ $DC21     ja
```

```
DBDE   20 25 D1   JSR $D125     Dateityp prüfen
DBE1   C9 04      CMP #$04     Rel-Datei ?
DBE3   F0 44      BEQ $DC29    ja
DBE5   B1 86      LDA ($86),Y
DBE7   29 8F      AND #$8F     Bit 4, 5 und 6 löschen
DBE9   91 86      STA ($86),Y  im Dateityp
DBEB   C8         INY
DBEC   B1 86      LDA ($86),Y  Tracknummer
DBEE   85 80      STA $80
DBF0   84 71      STY $71
DBF2   A0 1B      LDY #$1B
DBF4   B1 86      LDA ($86),Y  Sektornummer der Datei beim Überschreiben
DBF6   48         PHA          merken
DBF7   88         DEY
DBF8   B1 86      LDA ($86),Y  Tracknummer beim Überschreiben
D8FA   D0 0A      BNE $DC06    gesetzt ?
DBFC   85 80      STA $80      Tracknummer setzen
DBFE   68         PLA
DBFF   85 81      STA $81      Sektornummer
DC01   A9 67      LDA #$67
DC03   20 45 E6   JSR $E645    67, 'illegal track or sector'
DC06   48         PHA
DC07   A9 00      LDA #$00
DC09   91 86      STA ($86),Y  Tracknummer
DC0B   C8         INY
DC0C   91 86      STA ($86),Y  und Sektornummer der Ersatzdatei löschen
DC0E   68         PLA
DC0F   A4 71      LDY $71
DC11   91 86      STA ($86),Y
DC13   C8         INY          Track- und Sektornummer der neuen Datei setzen
DC14   B1 86      LDA ($86),Y
DC16   85 81      STA $81
DC18   68         PLA
DC19   91 86      STA ($86),Y
DC1B   20 7D C8   JSR $C87D    alte Datei löschen
DC1E   4C 29 DC   JMP $DC29

DC21   B1 86      LDA ($86),Y  Dateityp holen
DC23   29 0F      AND #$0F     Bit 0 bis 3 isolieren
DC25   09 80      ORA #$80     Bit 7 für geschlossene Datei setzen
DC27   91 86      STA ($86),Y
DC29   AE 70 02   LDX $0270    Kanalnummer
DC2C   A0 1C      LDY #$1C
DC2E   B5 B5      LDA $B5,X    Blockzahl lo
DC30   91 86      STA ($86),Y  in Directoryeintrag
DC32   C8         INY
DC33   B5 BB      LDA $BB,X    und Blockzahl hi
DC35   91 86      STA ($86),Y  schreiben
DC37   68         PLA          Puffernummer
DC38   AA         TAX
DC39   A9 90      LDA #$90     Kode für 'Schreiben'
DC3B   05 7F      ORA $7F      Drivenummer
DC3D   20 90 D5   JSR $D590    Block schreiben
DC40   68         PLA
DC41   85 83      STA $83      Sekundäradresse
DC43   4C 07 D1   JMP $D107    Kanal zum Schreiben öffnen
```

```
*****************************  Block lesen, Puffer belegen
DC46   A9 01      LDA #$01
DC48   20 E2 D1   JSR $D1E2     Kanal und Puffer zum Lesen suchen
DC4B   20 B6 DC   JSR $DCB6     Zeiger setzen
DC4E   AD 4A 02   LDA $024A     Dateityp
DC51   48         PHA           merken
DC52   0A         ASL  A
DC53   05 7F      ORA $7F       Drivenummer
DC55   95 EC      STA $EC,X
DC57   20 9B D0   JSR $D09B     Block in Puffer lesen
DC5A   A6 82      LDX $82       Kanalnummer
DC5C   A5 80      LDA $80       Track
DC5E   D0 05      BNE $DC65     Folgetrack ?
DC60   A5 81      LDA $81       Sektor
DC62   9D 44 02   STA $0244,X   als Endezeiger
DC65   68         PLA           Dateityp
DC66   C9 04      CMP #$04      Rel-Datei ?
DC68   D0 3F      BNE $DCA9     nein
DC6A   A4 83      LDY $83       Sekundäradresse
DC6C   B9 2B 02   LDA $022B,Y   Kanalnummer
DC6F   09 40      ORA #$40
DC71   99 2B 02   STA $022B,Y   Flag für READ und WRITE setzen
DC74   AD 58 02   LDA $0258     Recordlänge
DC77   95 C7      STA $C7,X
DC79   20 8E D2   JSR $D28E     Puffer für Side Sektor suchen
DC7C   10 03      BPL $DC81     gefunden ?
DC7E   4C 0F D2   JMP $D20F     70, 'no channel'

DC81   A6 82      LDX $82       Kanalnummer
DC83   95 CD      STA $CD,X
DC85   AC 59 02   LDY $0259
DC88   84 80      STY $80       Track für Side Sektor
DC8A   AC 5A 02   LDY $025A
DC8D   84 81      STY $81       Sektor für Side Sektor
DC8F   20 D3 D6   JSR $D6D3     Paramater an Disk-Controller übergeben
DC92   20 73 DE   JSR $DE73     Block lesen
DC95   20 99 D5   JSR $D599     und prüfen
DC98   A6 82      LDX $82       Kanalnummer
DC9A   A9 02      LDA #$02
DC9C   95 C1      STA $C1,X     Zeiger für Schreiben
DC9E   A9 00      LDA #$00
DCA0   20 C8 D4   JSR $D4C8     Pufferzeiger auf Null
DCA3   20 53 E1   JSR $E153     nächsten Record suchen
DCA6   4C 3E DE   JMP $DE3E     Track und Sektornummer holen

DCA9   20 56 D1   JSR $D156     Byte aus Puffer holen
DCAC   A6 82      LDX $82       Kanalnummer
DCAE   9D 3E 02   STA $023E,X   Byte ins Ausgaberegister
DCB1   A9 88      LDA #$88      Flag für READ setzen
DCB3   95 F2      STA $F2,X
DCB5   60         RTS

*****************************  Zeiger rücksetzen
DCB6   A6 82      LDX $82       Kanalnummer
DCB8   B5 A7      LDA $A7,X     Puffernummer
DCBA   0A         ASL  A        mal 2
```

```
DCBB   A8         TAY
DCBC   A9 02      LDA #$02
DCBE   99 99 00   STA $0099,Y   Pufferzeiger lo
DCC1   B5 AE      LDA $AE,X
DCC3   09 80      ORA #$80      Bit 7 setzen , Puffer nicht belegt
DCC5   95 AE      STA $AE,X
DCC7   0A         ASL  A
DCC8   A8         TAY
DCC9   A9 02      LDA #$02
DCCB   99 99 00   STA $0099,Y   Pufferzeiger lo
DCCE   A9 00      LDA #$00
DCD0   95 B5      STA $B5,X     Blockzahl lo
DCD2   95 BB      STA $BB,X     Blockzahl hi
DCD4   A9 00      LDA #$00
DCD6   9D 44 02   STA $0244,X   Endezeiger
DCD9   60         RTS

******************************  neuen Block anlegen
DCDA   20 A9 F1   JSR $F1A9     freien Sektor in BAM suchen
DCDD   A9 01      LDA #$01
DCDF   20 DF D1   JSR $D1DF     Kanal öffnen und Puffer belegen
DCE2   20 D0 D6   JSR $D6D0     Parameter an Disk-Controller übergeben
DCE5   20 B6 DC   JSR $DCB6     Zeiger rücksetzen
DCE8   A6 82      LDX $82       Kanalnummer
DCEA   AD 4A 02   LDA $024A     Dateityp
DCED   48         PHA
DCEE   0A         ASL  A
DCEF   05 7F      ORA $7F       Drivenummer
DCF1   95 EC      STA $EC,X     als Flag merken
DCF3   68         PLA
DCF4   C9 04      CMP #$04      Rel-Datei ?
DCF6   F0 05      BEQ $DCFD     ja
DCF8   A9 01      LDA #$01
DCFA   95 F2      STA $F2,X     WRITE-Flag setzen
DCFC   60         RTS

DCFD   A4 83      LDY $83       Sekundäradresse
DCFF   B9 2B 02   LDA $022B,Y   Kanalnummer in Tabelle
DD02   29 3F      AND #$3F      die obersten 2 Bit löschen
DD04   09 40      ORA #$40      Bit 6 setzen,
DD06   99 2B 02   STA $022B,Y   READ und WRITE Flag
DD09   AD 58 02   LDA $0258     Recordlänge
DD0C   95 C7      STA $C7,X     in Tabelle
DD0E   20 8E D2   JSR $D28E     Puffer suchen und belegen
DD11   10 03      BPL $DD16     gefunden ?
DD13   4C 0F D2   JMP $D20F     70, 'no channel'

DD16   A6 82      LDX $82       Kanalnummer
DD18   95 CD      STA $CD,X     Puffernummer für Side-Sektor
DD1A   20 C1 DE   JSR $DEC1     Puffer löschen
DD1D   20 1E F1   JSR $F11E     freien Block in BAM suchen
DD20   A5 80      LDA $80       Track
DD22   8D 59 02   STA $0259     für ersten Side-Sektor
DD25   A5 81      LDA $81       Sektor
DD27   8D 5A 02   STA $025A     für Side-Sektor
DD2A   A6 82      LDX $82       Kanalnummer
```

```
DD2C   B5 CD      LDA $CD,X     Puffernummer
DD2E   20 D3 D6   JSR $D6D3     Parameter an Disk-Controller übergeben
DD31   A9 00      LDA #$00
DD33   20 E9 DE   JSR $DEE9     Pufferzeiger auf Null
DD36   A9 00      LDA #$00
DD38   20 8D DD   JSR $DD8D     Null als Trackzeiger in Puffer
DD3B   A9 11      LDA #$11      17
DD3D   20 8D DD   JSR $DD8D     als Endezeiger in Puffer
DD40   A9 00      LDA #$00      Null
DD42   20 8D DD   JSR $DD8D     als Side-Sektor Nummer in Puffer
DD45   AD 58 02   LDA $0258     Recordlänge
DD48   20 8D DD   JSR $DD8D     in Puffer
DD4B   A5 80      LDA $80       Tracknummer dieses Blocks
DD4D   20 8D DD   JSR $DD8D     in Puffer
DD50   A5 81      LDA $81       Sektornummer
DD52   20 8D DD   JSR $DD8D     in Puffer
DD55   A9 10      LDA #$10      16
DD57   20 E9 DE   JSR $DEE9     Pufferzeiger auf 16
DD5A   20 3E DE   JSR $DE3E     Track und Sektornummer holen
DD5D   A5 80      LDA $80       Tracknummer des ersten Datenblocks
DD5F   20 8D DD   JSR $DD8D     in Puffer
DD62   A5 81      LDA $81       Sektornummer des ersten Datenblocks
DD64   20 8D DD   JSR $DD8D     in Puffer
DD67   20 6C DE   JSR $DE6C     Block auf Diskette schreiben
DD6A   20 99 D5   JSR $D599     und prüfen
DD6D   A9 02      LDA #$02
DD6F   20 C8 D4   JSR $D4C8     Pufferzeiger auf 2
DD72   A6 82      LDX $82       Kanalnummer
DD74   38         SEC
DD75   A9 00      LDA #$00
DD77   F5 C7      SBC $C7,X     Recordlänge
DD79   95 C1      STA $C1,X     Zeiger für Schreiben
DD7B   20 E2 E2   JSR $E2E2     Puffer löschen
DD7E   20 19 DE   JSR $DE19     Linkbytes in Puffer schreiben
DD81   20 5E DE   JSR $DE5E     Block auf Diskette schreiben
DD84   20 99 D5   JSR $D599     und prüfen
DD87   20 F4 EE   JSR $EEF4     BAM schreiben
DD8A   4C 98 DC   JMP $DC98     und fertig

******************************  Byte in Side-Sektor Block schreiben
DD8D   48         PHA           Byte merken
DD8E   A6 82      LDX $82       Kanalnummer
DD90   B5 CD      LDA $CD,X     Puffernummer des Side-Sektors
DD92   4C FD CF   JMP $CFFD     Byte in Puffer schreiben

******************************  Flags manipulieren
DD95   90 06      BCC $DD9D
DD97   A6 82      LDX $82       Kanalnummer
DD99   15 EC      ORA $EC,X     Flag setzen
DD9B   D0 06      BNE $DDA3
DD9D   A6 82      LDX $82       Kanalnummer
DD9F   49 FF      EOR #$FF
DDA1   35 EC      AND $EC,X     Flag löschen
DDA3   95 EC      STA $EC,X
DDA5   60         RTS
```

```
DDA6   A6 82      LDX $82        Kanalnummer
DDA8   35 EC      AND $EC,X      Flag testen
DDAA   60         RTS

****************************** Befehlskode für Schreiben prüfen
DDAB   20 93 DF   JSR $DF93      Puffernummer holen
DDAE   AA         TAX
DDAF   BD 5B 02   LDA $025B,X
DDB2   29 F0      AND #$F0       Befehlskode isolieren
DDB4   C9 90      CMP #$90       Kode für schreiben ?
DDB6   60         RTS

******************************
DDB7   A2 00      LDX #$00
DDB9   86 71      STX $71        Zähler für Sekundäradressen
DDBB   BD 2B 02   LDA $022B,X    Kanalnummer aus Tabelle holen
DDBE   C9 FF      CMP #$FF
DDC0   D0 08      BNE $DDCA      Datei offen ?
DDC2   A6 71      LDX $71
DDC4   E8         INX            Zähler erhöhen
DDC5   E0 10      CPX #$10       noch kleiner als 16 ?
DDC7   90 F0      BCC $DDB9
DDC9   60         RTS

DDCA   86 71      STX $71
DDCC   29 3F      AND #$3F       Kanalnummer isolieren
DDCE   A8         TAY
DDCF   B9 EC 00   LDA $00EC,Y
DDD2   29 01      AND #$01       Drivenummer isolieren
DDD4   85 70      STA $70
DDD6   AE 53 02   LDX $0253
DDD9   B5 E2      LDA $E2,X
DDDB   29 01      AND #$01       Drivenummer isolieren
DDDD   C5 70      CMP $70        gleiches Drive ?
DDDF   D0 E1      BNE $DDC2      nein
DDE1   B9 60 02   LDA $0260,Y    Sektornummer im Directory
DDE4   D5 D8      CMP $D8,X      gleiche wie Datei ?
DDE6   D0 DA      BNE $DDC2      nein
DDE8   B9 66 02   LDA $0266,Y
DDEB   D5 DD      CMP $DD,X      Zeiger gleich ?
DDED   D0 D3      BNE $DDC2      nein
DDEF   18         CLC
DDF0   60         RTS

****************************** Block einer Rel-Datei schreiben
DDF1   20 9E DF   JSR $DF9E      Puffernummer holen
DDF4   50 06      BVC $DDFC      keine Rel-Datei ?
DDF6   20 5E DE   JSR $DE5E      Block schreiben
DDF9   20 99 D5   JSR $D599      und prüfen
DDFC   60         RTS

****************************** Bytes für Folgetrack schreiben
DDFD   20 2B DE   JSR $DE2B      Pufferzeiger setzen
DE00   A5 80      LDA $80        Tracknummer
DE02   91 94      STA ($94),Y    in Puffer
DE04   C8         INY
```

```
DE05   A5 81      LDA $81       Sektornummer
DE07   91 94      STA ($94),Y   in Puffer
DE09   4C 05 E1   JMP $E105     Rel-Flag setzen

****************************  Folgetrack und Sektornummer holen
DE0C   20 2B DE   JSR $DE2B     Pufferzeiger setzen
DE0F   B1 94      LDA ($94),Y   Folgetracknummer
DE11   85 80      STA $80
DE13   C8         INY
DE14   B1 94      LDA ($94),Y   und Sektornummer holen
DE16   85 81      STA $81
DE18   60         RTS

****************************  Folgetrack bei letztem Block
DE19   20 2B DE   JSR $DE2B     Pufferzeiger setzen
DE1C   A9 00      LDA #$00      null
DE1E   91 94      STA ($94),Y   als Tracknummer
DE20   C8         INY
DE21   A6 82      LDX $82       Kanalnummer
DE23   B5 C1      LDA $C1,X     Zeiger in Block
DE25   AA         TAX
DE26   CA         DEX           minus 1
DE27   8A         TXA
DE28   91 94      STA ($94),Y   als Zeiger in Block
DE2A   60         RTS

****************************  Pufferzeiger auf Null
DE2B   20 93 DF   JSR $DF93     Puffernummer holen
DE2E   0A         ASL  A        mal 2
DE2F   AA         TAX
DE30   B5 9A      LDA $9A,X     Pufferzeiger hi
DE32   85 95      STA $95
DE34   A9 00      LDA #$00
DE36   85 94      STA $94       Pufferzeiger lo
DE38   A0 00      LDY #$00
DE3A   60         RTS
****************************  Track und Sektor holen
DE3B   20 EB D0   JSR $D0EB     Kanalnummer holen
DE3E   20 93 DF   JSR $DF93     Puffernummer holen
DE41   85 F9      STA $F9       merken
DE43   0A         ASL  A        mal 2
DE44   A8         TAY
DE45   B9 06 00   LDA $0006,Y   Track
DE48   85 80      STA $80
DE4A   B9 07 00   LDA $0007,Y   und Sektor vom Disk-Controller holen
DE4D   85 81      STA $81
DE4F   60         RTS

****************************
DE50   A9 90      LDA #$90      Befehlskode für Schreiben
DE52   8D 4D 02   STA $024D
DE55   D0 28      BNE $DE7F

DE57   A9 80      LDA #$80      Befehlskode für Lesen
DE59   8D 4D 02   STA $024D
DE5C   D0 21      BNE $DE7F
```

```
DE5E   A9 90      LDA #$90      Befehlskode für Schreiben
DE60   8D 4D 02   STA $024D
DE63   D0 26      BNE $DE8B

DE65   A9 80      LDA #$80      Befehlskode für Lesen
DE67   8D 4D 02   STA $024D
DE6A   D0 1F      BNE $DE8B

DE6C   A9 90      LDA #$90      Befehlskode für Schreiben
DE6E   8D 4D 02   STA $024D
DE71   D0 02      BNE $DE75

DE73   A9 80      LDA #$80      Befehlskode für Lesen
DE75   8D 4D 02   STA $024D
DE78   A6 82      LDX $82       Kanalnummer
DE7A   B5 CD      LDA $CD,X     Puffernummer Side-Sektor
DE7C   AA         TAX
DE7D   10 13      BPL $DE92     Puffer zugeordnet ?
DE7F   20 D0 D6   JSR $D6D0     Header für Disk-Controller generieren
DE82   20 93 DF   JSR $DF93     Puffernummer holen
DE85   AA         TAX
DE86   A5 7F      LDA $7F       Drivenummer
DE88   9D 5B 02   STA $025B,X
DE8B   20 15 E1   JSR $E115     Puffernummer
DE8E   20 93 DF   JSR $DF93     Puffernummer holen
DE91   AA         TAX
DE92   4C 06 D5   JMP $D506     Block schreiben

****************************** Folgetrack und Sektor aus Puffer holen
DE95   A9 00      LDA #$00
DE97   20 C8 D4   JSR $D4C8     Pufferzeiger auf Null
DE9A   20 37 D1   JSR $D137     Byte holen
DE9D   85 80      STA $80       als Track merken
DE9F   20 37 D1   JSR $D137     Byte holen
DEA2   85 81      STA $81       als Sektor
DEA4   60         RTS

****************************** Pufferinhalte kopieren
DEA5   48         PHA
DEA6   A9 00      LDA #$00
DEA8   85 6F      STA $6F
DEAA   85 71      STA $71
DEAC   B9 E0 FE   LDA $FEE0,Y   Pufferadresse Y, Hi
DEAF   85 70      STA $70
DEB1   BD E0 FE   LDA $FEE0,X   Pufferadresse X, Hi
DEB4   85 72      STA $72
DEB6   68         PLA
DEB7   A8         TAY
DEB8   88         DEY
DEB9   B1 6F      LDA ($6F),Y   Inhalt Puffer Y
DEBB   91 71      STA ($71),Y   nach Puffer X kopieren
DEBD   88         DEY
DEBE   10 F9      BPL $DEB9
DEC0   60         RTS

****************************** Puffer Y löschen
```

```
DEC1   A8          TAY            Puffernummer
DEC2   B9 E0 FE    LDA $FEE0,Y    Hi-Adresse holen
DEC5   85 70       STA $70
DEC7   A9 00       LDA #$00       Lo-Adresse
DEC9   85 6F       STA $6F
DECB   A8          TAY
DECC   91 6F       STA ($6F),Y    Puffer löschen
DECE   C8          INY
DECF   D0 FB       BNE $DECC
DED1   60          RTS

******************************  Side-Sektor Nummer holen
DED2   A9 00       LDA #$00
DED4   20 DC DE    JSR $DEDC      Pufferzeiger auf null
DED7   A0 02       LDY #$02
DED9   B1 94       LDA ($94),Y    Byte 2 enthält Side-Sektor Nummer
DEDB   60          RTS

******************************  Pufferzeiger auf Side-Sektor setzen
DEDC   85 94       STA $94        Zeiger lo
DEDE   A6 82       LDX $82        Kanalnummer
DEE0   B5 CD       LDA $CD,X      Puffernummer
DEE2   AA          TAX
DEE3   BD E0 FE    LDA $FEE0,X    Pufferadresse Hi
DEE6   85 95       STA $95        setzen
DEE8   60          RTS

******************************  Pufferzeiger für Side-Sektor
DEE9   48          PHA            Zeiger in Side-Sektor
DEEA   20 DC DE    JSR $DEDC      Pufferzeiger setzen
DEED   48          PHA
DEEE   8A          TXA            Puffernummer
DEEF   0A          ASL  A         mal 2
DEF0   AA          TAX
DEF1   68          PLA            Pufferzeiger hi
DEF2   95 9A       STA $9A,X
DEF4   68          PLA            Pufferzeiger lo
DEF5   95 99       STA $99,X
DEF7   60          RTS

******************************  Side-Sektor und Pufferzeiger holen
DEF8   20 66 DF    JSR $DF66      ist Side-Sektor im Puffer ?
DEFB   30 0E       BMI $DF0B      nein
DEFD   50 13       BVC $DF12      ok
DEFF   A6 82       LDX $82        Kanalnummer
DF01   B5 CD       LDA $CD,X      Puffernummer
DF03   20 1B DF    JSR $DF1B      Side-Sektor lesen
DF06   20 66 DF    JSR $DF66      und prüfen ob im Puffer
DF09   10 07       BPL $DF12      ja ?
DF0B   20 CB E1    JSR $E1CB      letzten Side-Sektor holen
DF0E   2C CE FE    BIT $FECE      V-Bit setzen
DF11   60          RTS
DF12   A5 D6       LDA $D6        Side-Sektor Endezeiger
DF14   20 E9 DE    JSR $DEE9      Zeiger in Side-Sektor setzen
DF17   2C CD FE    BIT $FECD      V-Bit löschen
DF1A   60          RTS
```

```
******************************  Side-Sektor lesen
DF1B   85 F9      STA $F9       Puffernummer
DF1D   A9 80      LDA #$80      Befehlskode für lesen
DF1F   D0 04      BNE $DF25

******************************  Side-Sektor schreiben
DF21   85 F9      STA $F9       Puffernummer
DF23   A9 90      LDA #$90      Befehlskode für schreiben
DF25   48         PHA
DF26   B5 EC      LDA $EC,X
DF28   29 01      AND #$01      Drivenummer isolieren
DF2A   85 7F      STA $7F
DF2C   68         PLA
DF2D   05 7F      ORA $7F       Befehlskode plus Drivenummer
DF2F   8D 4D 02   STA $024D     merken
DF32   B1 94      LDA ($94),Y   Tracknummer
DF34   85 80      STA $80
DF36   C8         INY
DF37   B1 94      LDA ($94),Y   Sektornummer
DF39   85 81      STA $81
DF3B   A5 F9      LDA $F9       Puffernummer
DF3D   20 D3 D6   JSR $D6D3     Parameter an Disk-Controller übergeben
DF40   A6 F9      LDX $F9       Puffernummer
DF42   4C 93 D5   JMP $D593     Befehl an Disk-Controller übergeben

******************************  Pufferzeiger in Side-Sektor setzen
DF45   A6 82      LDX $82       Kanalnummer
DF47   B5 CD      LDA $CD,X     Puffernummer
DF49   4C EB D4   JMP $D4EB     Pufferzeiger setzen

******************************  Blockzahl einer Rel-Datei berechnen
DF4C   A9 78      LDA #$78      120 Blockzeiger pro Sidesektor
DF4E   20 5C DF   JSR $DF5C     zu $70/$71 addieren
DF51   CA         DEX           Side-Sektornummer
DF52   10 F8      BPL $DF4C     nächster Side-Sektor ?
DF54   A5 72      LDA $72       Anzahl der Zeiger im letzten Block
DF56   4A         LSR  A        durch 2
DF57   20 5C DF   JSR $DF5C     zu bisheriger Summe addieren
DF5A   A5 73      LDA $73       Zahl der belegten Side-Sektor Blöcke
DF5C   18         CLC
DF5D   65 70      ADC $70
DF5F   85 70      STA $70       addieren
DF61   90 02      BCC $DF65
DF63   E6 71      INC $71
DF65   60         RTS

******************************  Side-Sektor im Puffer prüfen
DF66   20 D2 DE   JSR $DED2     Side-Sektor Nummer holen
DF69   C5 D5      CMP $D5       gleich Nummer des erforderlichen Blocks ?
DF6B   D0 0E      BNE $DF7B     nein
DF6D   A4 D6      LDY $D6       Zeiger in Side-Sektor
DF6F   B1 94      LDA ($94),Y   Tracknummer
DF71   F0 04      BEQ $DF77     noch nicht angelegt ?
DF73   2C CD FE   BIT $FECD     Bits löschen
DF76   60         RTS
```

```
DF77   2C CF FE   BIT $FECF      N-Bit setzen
DF7A   60         RTS

DF7B   A5 D5      LDA $D5        Side-Sektornummer
DF7D   C9 06      CMP #$06       6 oder größer ?
DF7F   B0 0A      BCS $DF8B      ja
DF81   0A         ASL  A
DF82   A8         TAY
DF83   A9 04      LDA #$04
DF85   85 94      STA $94
DF87   B1 94      LDA ($94),Y    Tracknummer
DF89   D0 04      BNE $DF8F      bereits angelegt ?
DF8B   2C D0 FE   BIT $FED0      N- und V-Bit setzen
DF8E   60         RTS

DF8F   2C CE FE   BIT $FECE      V-Bit setzen
DF92   60         RTS

****************************** Puffernummer holen
DF93   A6 82      LDX $82        Kanalnummer
DF95   B5 A7      LDA $A7,X      Puffernummer
DF97   10 02      BPL $DF9B      belegt ?
DF99   B5 AE      LDA $AE,X      Puffernummer aus zweiter Tabelle
DF9B   29 BF      AND #$BF       V-Bit löschen
DF9D   60         RTS

DF9E   A6 82      LDX $82        Kanalnummer
DFA0   8E 57 02   STX $0257      merken
DFA3   B5 A7      LDA $A7,X      Puffernummer holen
DFA5   10 09      BPL $DFB0      Puffer belegt ?
DFA7   8A         TXA
DFA8   18         CLC
DFA9   69 07      ADC #$07       Nummer um sieben erhöhen
DFAB   8D 57 02   STA $0257      und merken
DFAE   B5 AE      LDA $AE,X      Puffernummer aus Tabelle 2
DFB0   85 70      STA $70
DFB2   29 1F      AND #$1F       die obersten 3 Bit löschen
DFB4   24 70      BIT $70
DFB6   60         RTS

DFB7   A6 82      LDX $82        Kanalnummer
DFB9   B5 A7      LDA $A7,X      Puffernummer
DFBB   30 02      BMI $DFBF      Puffer frei ?
DFBD   B5 AE      LDA $AE,X      Puffernummer aus Tabelle 2
DFBF   C9 FF      CMP #$FF       frei ?
DFC1   60         RTS

DFC2   A6 82      LDX $82
DFC4   09 80      ORA #$80
DFC6   B4 A7      LDY $A7,X
DFC8   10 03      BPL $DFCD
DFCA   95 A7      STA $A7,X
DFCC   60         RTS
DFCD   95 AE      STA $AE,X
DFCF   60         RTS
```

```
*****************************  nächsten Record in Rel-Datei holen
DFD0   A9 20      LDA #$20
DFD2   20 9D DD   JSR $DD9D     Bit 5 löschen
DFD5   A9 80      LDA #$80
DFD7   20 A6 DD   JSR $DDA6     Bit 7 testen
DFDA   D0 41      BNE $E01D     gesetzt ?
DFDC   A6 82      LDX $82       Kanalnummer
DFDE   F6 B5      INC $B5,X     Recordnummer erhöhen
DFE0   D0 02      BNE $DFE4
DFE2   F6 BB      INC $BB,X     Recordnummer hi
DFE4   A6 82      LDX $82       Kanalnummer
DFE6   B5 C1      LDA $C1,X     Schreibzeiger
DFE8   F0 2E      BEQ $E018     null ?
DFEA   20 E8 D4   JSR $D4E8     Pufferzeiger setzen
DFED   A6 82      LDX $82       Kanalnummer
DFEF   D5 C1      CMP $C1,X     Pufferzeiger kleiner Schreibzeiger ?
DFF1   90 03      BCC $DFF6     ja
DFF3   20 3C E0   JSR $E03C     Block schreiben, nächsten Block lesen
DFF6   A6 82      LDX $82       Kanalnummer
DFF8   B5 C1      LDA $C1,X     Schreibzeiger
DFFA   20 C8 D4   JSR $D4C8     Pufferzeiger gleich Schreibzeiger setzen
DFFD   A1 99      LDA ($99,X)   Byte aus Puffer
DFFF   85 85      STA $85       in Ausgaberegister holen
E001   A9 20      LDA #$20
E003   20 9D DD   JSR $DD9D     Bit 5 löschen
E006   20 04 E3   JSR $E304     Recordlänge zu Schreibzeiger addieren
E009   48        PHA           und merken
E00A   90 28      BCC $E034     noch nicht im nächsten Block ?
E00C   A9 00      LDA #$00
E00E   20 F6 D4   JSR $D4F6     Tracknummer holen
E011   D0 21      BNE $E034     Block vorhanden ?
E013   68        PLA           Zeiger
E014   C9 02      CMP #$02      gleich 2
E016   F0 12      BEQ $E02A     ja
E018   A9 80      LDA #$80
E01A   20 97 DD   JSR $DD97     Bit 7 setzen
E01D   20 2F D1   JSR $D12F     Byte aus Puffer holen
E020   B5 99      LDA $99,X     Pufferzeiger
E022   99 44 02   STA $0244,Y   als Endezeiger
E025   A9 0D      LDA #$0D      CR
E027   85 85      STA $85       in Ausgaberegister
E029   60        RTS

E02A   20 35 E0   JSR $E035
E02D   A6 82      LDX $82       Kanalnummer
E02F   A9 00      LDA #$00
E031   95 C1      STA $C1,X     Schreibzeiger auf null
E033   60        RTS

E034   68        PLA
E035   A6 82      LDX $82       Kanalnummer
E037   95 C1      STA $C1,X     Schreibzeiger setzen
E039   4C 6E E1   JMP $E16E

*****************************  Block schreiben und nächsten Block lesen
E03C   20 D3 D1   JSR $D1D3     Drivenummer holen
```

```
E03F   20 95 DE   JSR $DE95      Track und Sektornummer holen
E042   20 9E DF   JSR $DF9E      Puffernummer holen
E045   50 16      BVC $E05D      keine Rel-Datei ?
E047   20 5E DE   JSR $DE5E      Block schreiben
E04A   20 1E CF   JSR $CF1E      Puffer wechseln
E04D   A9 02      LDA #$02
E04F   20 C8 D4   JSR $D4C8      Pufferzeiger auf zwei
E052   20 AB DD   JSR $DDAB      Befehlskode für Schreiben ?
E055   D0 24      BNE $E07B      nein
E057   20 57 DE   JSR $DE57      Block lesen
E05A   4C 99 D5   JMP $D599      und prüfen

E05D   20 1E CF   JSR $CF1E      Puffer wechseln
E060   20 AB DD   JSR $DDAB      Befehlskode für Schreiben ?
E063   D0 06      BNE $E06B      nein
E065   20 57 DE   JSR $DE57      Block lesen
E068   20 99 D5   JSR $D599      und prüfen
E06B   20 95 DE   JSR $DE95      Track und Sektornummer holen
E06E   A5 80      LDA $80        Track
E070   F0 09      BEQ $E07B      kein Folgetrack ?
E072   20 1E CF   JSR $CF1E      Puffer wechseln
E075   20 57 DE   JSR $DE57      Block lesen
E078   20 1E CF   JSR $CF1E      Puffer wechseln
E07B   60         RTS

******************************  ein Byte in Record schreiben
E07C   20 05 E1   JSR $E105
E07F   20 93 DF   JSR $DF93      Puffernummer holen
E082   0A         ASL  A         mal 2
E083   AA         TAX
E084   A5 85      LDA $85        Datenbyte
E086   81 99      STA ($99,X)    in Puffer schreiben
E088   B4 99      LDY $99,X      Pufferzeiger
E08A   C8         INY            erhöhen
E08B   D0 09      BNE $E096      ungleich null ?
E08D   A4 82      LDY $82        Kanalnummer
E08F   B9 C1 00   LDA $00C1,Y    Schreibzeiger
E092   F0 0A      BEQ $E09E      gleich null ?
E094   A0 02      LDY #$02       Pufferzeiger auf 2
E096   98         TYA
E097   A4 82      LDY $82        Kanalnummer
E099   D9 C1 00   CMP $00C1,Y    Pufferzeiger gleich Schreibzeiger ?
E09C   D0 05      BNE $E0A3      nein
E09E   A9 20      LDA #$20
E0A0   4C 97 DD   JMP $DD97      Bit 5 setzen

E0A3   F6 99      INC $99,X      Pufferzeiger erhöhen
E0A5   D0 03      BNE $E0AA      ungleich null ?
E0A7   20 3C E0   JSR $E03C      sonst Block schreiben, nächsten Block lesen
E0AA   60         RTS

******************************  Byte in Rel-Datei schreiben
E0AB   A9 A0      LDA #$A0
E0AD   20 A6 DD   JSR $DDA6      Bit 6 und 7 testen
E0B0   D0 27      BNE $E0D9      gesetzt ?
E0B2   A5 85      LDA $85        Datenbyte
```

```
E0B4   20 7C E0   JSR $E07C      in Record schreiben
E0B7   A5 F8      LDA $F8        Ende ?
E0B9   F0 0D      BEQ $E0C8      ja
E0BB   60         RTS

E0BC   A9 20      LDA #$20
E0BE   20 A6 DD   JSR $DDA6      Bit 5 testen
E0C1   F0 05      BEQ $E0C8      nicht gesetzt
E0C3   A9 51      LDA #$51       51, 'overflow in record'
E0C5   8D 6C 02   STA $026C      Fehlerflag setzen
E0C8   20 F3 E0   JSR $E0F3      Rest des Records mit Nullen füllen
E0CB   20 53 E1   JSR $E153
E0CE   AD 6C 02   LDA $026C      Fehlerflag gesetzt ?
E0D1   F0 03      BEQ $E0D6      nein
E0D3   4C C8 C1   JMP $C1C8      Fehlermeldung setzen

E0D6   4C BC E6   JMP $E6BC      fehlerfreie Ausführung

E0D9   29 80      AND #$80       Bit 7 gesetzt ?
E0DB   D0 05      BNE $E0E2      ja
E0DD   A5 F8      LDA $F8
E0DF   F0 DB      BEQ $E0BC      Ende ?
E0E1   60         RTS

E0E2   A5 85      LDA $85        Datenbyte
E0E4   48         PHA
E0E5   20 1C E3   JSR $E31C      Side-Sektoren erweitern
E0E8   68         PLA
E0E9   85 85      STA $85
E0EB   A9 80      LDA #$80
E0ED   20 9D DD   JSR $DD9D      Bit 7 löschen
E0F0   4C B2 E0   JMP $E0B2      Byte in Datei schreiben

******************************   Record mit Nullen auffüllen
E0F3   A9 20      LDA #$20
E0F5   20 A6 DD   JSR $DDA6      Bit 5 testen
E0F8   D0 0A      BNE $E104      gesetzt ?
E0FA   A9 00      LDA #$00
E0FC   85 85      STA $85        null als Datenbyte
E0FE   20 7C E0   JSR $E07C      in Record schreiben
E101   4C F3 E0   JMP $E0F3      bis Record voll

E104   60         RTS

******************************   Puffernummer in Tabelle schreiben
E105   A9 40      LDA #$40
E107   20 97 DD   JSR $DD97      Bit 6 setzen
E10A   20 9E DF   JSR $DF9E      Puffernummer holen
E10D   09 40      ORA #$40       Bit 6 setzen
E10F   AE 57 02   LDX $0257      Kanalnummer + 7
E112   95 A7      STA $A7,X      in Tabelle schreiben
E114   60         RTS

E115   20 9E DF   JSR $DF9E      Puffernummer holen
E118   29 BF      AND #$BF       Bit 6 löschen
E11A   AE 57 02   LDX $0257      Kanalnummer + 7
```

```
E11D   95 A7      STA $A7,X     in Tabelle schreiben
E11F   60         RTS

****************************  Byte aus Rel-Datei holen
E120   A9 80      LDA #$80
E122   20 A6 DD   JSR $DDA6     Bit 7 testen
E125   D0 37      BNE $E15E     gesetzt ?
E127   20 2F D1   JSR $D12F     Byte aus Puffer holen
E12A   B5 99      LDA $99,X     Pufferzeiger
E12C   D9 44 02   CMP $0244,Y   mit Endezeiger vergleichen
E12F   F0 22      BEQ $E153     gleich ?
E131   F6 99      INC $99,X     Pufferzeiger erhöhen
E133   D0 06      BNE $E13B     ungleich null ?
E135   20 3C E0   JSR $E03C     Block schreiben, nächsten Block lesen
E138   20 2F D1   JSR $D12F     Byte aus Puffer holen
E13B   A1 99      LDA ($99,X)
E13D   99 3E 02   STA $023E,Y   ins Ausgaberegister
E140   A9 89      LDA #$89
E142   99 F2 00   STA $00F2,Y   READ und WRITE Flag setzen
E145   B5 99      LDA $99,X     Pufferzeiger
E147   D9 44 02   CMP $0244,Y   mit Endezeiger vergleichen
E14A   F0 01      BEQ $E14D     gleich ?
E14C   60         RTS

E14D   A9 81      LDA #$81
E14F   99 F2 00   STA $00F2,Y   Flag für Ende setzen
E152   60         RTS

E153   20 D0 DF   JSR $DFD0     nächsten Record suchen
E156   20 2F D1   JSR $D12F     Puffernummer und Kanalnummer holen
E159   A5 85      LDA $85       Datenbyte
E15B   4C 3D E1   JMP $E13D     ins Ausgaberegister

E15E   A6 82      LDX $82       Kanalnummer
E160   A9 0D      LDA #$0D      CR
E162   9D 3E 02   STA $023E,X   ins Ausgaberegister
E165   A9 81      LDA #$81
E167   95 F2      STA $F2,X     Flag für Ende setzen
E169   A9 50      LDA #$50
E16B   20 C8 C1   JSR $C1C8     50, 'record not present'

E16E   A6 82      LDX $82       Kanalnummer
E170   B5 C1      LDA $C1,X     Schreibzeiger
E172   85 87      STA $87       merken
E174   C6 87      DEC $87
E176   C9 02      CMP #$02      gleich 2 ?
E178   D0 04      BNE $E17E     nein
E17A   A9 FF      LDA #$FF
E17C   85 87      STA $87
E17E   B5 C7      LDA $C7,X     Recordlänge
E180   85 88      STA $88
E182   20 E8 D4   JSR $D4E8     Pufferzeiger setzen
E185   A6 82      LDX $82       Kanalnummer
E187   C5 87      CMP $87       Pufferzeiger größer als Schreibzeiger ?
E189   90 19      BCC $E1A4
E18B   F0 17      BEQ $E1A4     nein
```

```
E18D   20 1E CF   JSR $CF1E     Puffer wechseln
E190   20 B2 E1   JSR $E1B2
E193   90 08      BCC $E19D
E195   A6 82      LDX $82       Kanalnummer
E197   9D 44 02   STA $0244,X
E19A   4C 1E CF   JMP $CF1E     Puffer wechseln

E19D   20 1E CF   JSR $CF1E     Puffer wechseln
E1A0   A9 FF      LDA #$FF
E1A2   85 87      STA $87
E1A4   20 B2 E1   JSR $E1B2
E1A7   B0 03      BCS $E1AC
E1A9   20 E8 D4   JSR $D4E8     Pufferzeiger setzen
E1AC   A6 82      LDX $82       Kanalnummer
E1AE   9D 44 02   STA $0244,X   Endezeiger
E1B1   60         RTS

E1B2   20 2B DE   JSR $DE2B     Pufferzeiger auf null
E1B5   A4 87      LDY $87
E1B7   B1 94      LDA ($94),Y   Byte aus Puffer
E1B9   D0 0D      BNE $E1C8     ungleich null ?
E1BB   88         DEY
E1BC   C0 02      CPY #$02
E1BE   90 04      BCC $E1C4
E1C0   C6 88      DEC $88
E1C2   D0 F3      BNE $E1B7
E1C4   C6 88      DEC $88
E1C6   18         CLC
E1C7   60         RTS

E1C8   98         TYA
E1C9   38         SEC
E1CA   60         RTS

****************************  letzten Side-Sektor holen
E1CB   20 D2 DE   JSR $DED2     Nummer des Side-Sektors holen
E1CE   85 D5      STA $D5       merken
E1D0   A9 04      LDA #$04
E1D2   85 94      STA $94       Zeiger auf Side-Sektoren
E1D4   A0 0A      LDY #$0A
E1D6   D0 04      BNE $E1DC

E1D8   88         DEY
E1D9   88         DEY
E1DA   30 26      BMI $E202
E1DC   B1 94      LDA ($94),Y   Tracknummer der vorhergehende Blocks
E1DE   F0 F8      BEQ $E1D8     noch nicht angelegt ?
E1E0   98         TYA
E1E1   4A         LSR  A        durch 2 ergibt Nummer
E1E2   C5 D5      CMP $D5       gleich Nummer des aktuellen Blocks ?
E1E4   F0 09      BEQ $E1EF     ja
E1E6   85 D5      STA $D5       sonst als Nummer merken
E1E8   A6 82      LDX $82       Kanalnummer
E1EA   B5 CD      LDA $CD,X     Puffernummer
E1EC   20 1B DF   JSR $DF1B     Block lesen
E1EF   A0 00      LDY #$00
```

```
E1F1   84 94      STY $94        Pufferzeiger
E1F3   B1 94      LDA ($94),Y    Tracknummer
E1F5   D0 0B      BNE $E202      folgt weiterer Block ?
E1F7   C8         INY
E1F8   B1 94      LDA ($94),Y    Sektornummer gleich Endezeiger
E1FA   A8         TAY
E1FB   88         DEY
E1FC   84 D6      STY $D6        Endezeiger merken
E1FE   98         TYA
E1FF   4C E9 DE   JMP $DEE9      Pufferzeiger setzen

E202   A9 67      LDA #$67
E204   20 45 E6   JSR $E645      67, 'illegal track or sector'

****************************** P-Befehl,   'Record'
E207   20 B3 C2   JSR $C2B3      Zeile prüfen
E20A   AD 01 02   LDA $0201      Sekundäradresse
E20D   85 83      STA $83
E20F   20 EB D0   JSR $D0EB      sucht Kanalnummer
E212   90 05      BCC $E219      gefunden ?
E214   A9 70      LDA #$70
E216   20 C8 C1   JSR $C1C8      70, 'no channel'

E219   A9 A0      LDA #$A0
E21B   20 9D DD   JSR $DD9D      Bit 6 und 7 löschen
E21E   20 25 D1   JSR $D125      prüft auf 'REL'-File
E221   F0 05      BEQ $E228      ja
E223   A9 64      LDA #$64
E225   20 C8 C1   JSR $C1C8      64, 'file type mismatch'

E228   B5 EC      LDA $EC,X
E22A   29 01      AND #$01
E22C   85 7F      STA $7F        Drivenummer
E22E   AD 02 02   LDA $0202      Recordnummer lo
E231   95 B5      STA $B5,X
E233   AD 03 02   LDA $0203      Recordnummer hi
E236   95 BB      STA $BB,X
E238   A6 82      LDX $82        Kanalnummer
E23A   A9 89      LDA #$89
E23C   95 F2      STA $F2,X      READ und WRITE-Flag
E23E   AD 04 02   LDA $0204      Byte-Pointer
E241   F0 10      BEQ $E253      null ?
E243   38         SEC
E244   E9 01      SBC #$01
E246   F0 0B      BEQ $E253
E248   D5 C7      CMP $C7,X      mit Recordlänge des Files vergleichen
E24A   90 07      BCC $E253
E24C   A9 51      LDA #$51
E24E   8D 6C 02   STA $026C      51, 'overflow in record'
E251   A9 00      LDA #$00
E253   85 D4      STA $D4
E255   20 0E CE   JSR $CE0E      Zeiger in Rel-Datei berechnen
E258   20 F8 DE   JSR $DEF8      und entsprechenden Side-Sektor-Block lesen
E25B   50 08      BVC $E265      Block vorhanden ?
E25D   A9 80      LDA #$80
E25F   20 97 DD   JSR $DD97      Bit 7 setzen
```

```
E262   4C 5E E1   JMP $E15E     und 50, 'record not present'

E265   20 75 E2   JSR $E275
E268   A9 80      LDA #$80
E26A   20 A6 DD   JSR $DDA6     Bit 7 testen
E26D   F0 03      BEQ $E272     nicht gesetzt
E26F   4C 5E E1   JMP $E15E     50, 'record not present'
E272   4C 94 C1   JMP $C194     fertig

E275   20 9C E2   JSR $E29C
E278   A5 D7      LDA $D7       Zeiger in Rel-Datei
E27A   20 C8 D4   JSR $D4C8     Pufferzeiger setzen
E27D   A6 82      LDX $82       Kanalnummer
E27F   B5 C7      LDA $C7,X     Recordlänge
E281   38         SEC
E282   E5 D4      SBC $D4       minus Position
E284   B0 03      BCS $E289     positiv ?
E286   4C 02 E2   JMP $E202     67, 'illegal track or sector'

E289   18         CLC
E28A   65 D7      ADC $D7       Zeiger in Datenblock addieren
E28C   90 03      BCC $E291     kein Überlauf
E28E   69 01      ADC #$01      plus 2
E290   38         SEC
E291   20 09 E0   JSR $E009     Zeiger setzen
E294   4C 38 E1   JMP $E138     und Byte aus Puffer holen

E297   A9 51      LDA #$51
E299   20 C8 C1   JSR $C1C8     51, 'overflow in record'

E29C   A5 94      LDA $94       Pufferzeiger lo
E29E   85 89      STA $89
E2A0   A5 95      LDA $95       Pufferzeiger hi
E2A2   85 8A      STA $8A
E2A4   20 D0 E2   JSR $E2D0     Track und Sektor vergleichen
E2A7   D0 01      BNE $E2AA     ungleich ?
E2A9   60         RTS

E2AA   20 F1 DD   JSR $DDF1
E2AD   20 0C DE   JSR $DE0C
E2B0   A5 80      LDA $80       Track
E2B2   F0 0E      BEQ $E2C2     kein Folgeblock ?
E2B4   20 D3 E2   JSR $E2D3     Track und Sektornummer vergleichen
E2B7   D0 06      BNE $E2BF     ungleich ?
E2B9   20 1E CF   JSR $CF1E     Puffer wechseln
E2BC   4C DA D2   JMP $D2DA
E2BF   20 DA D2   JSR $D2DA
E2C2   A0 00      LDY #$00
E2C4   B1 89      LDA ($89),Y   Track
E2C6   85 80      STA $80
E2C8   C8         INY
E2C9   B1 89      LDA ($89),Y   und Sektor des nächsten Blocks
E2CB   85 81      STA $81
E2CD   4C AF D0   JMP $D0AF     Block lesen

E2D0   20 3E DE   JSR $DE3E
```

```
E2D3   A0 00      LDY #$00
E2D5   B1 89      LDA ($89),Y   Tracknummer
E2D7   C5 80      CMP $80       vergleichen
E2D9   F0 01      BEQ $E2DC
E2DB   60         RTS
E2DC   C8         INY
E2DD   B1 89      LDA ($89),Y   Sektornummer
E2DF   C5 81      CMP $81       vergleichen
E2E1   60         RTS

****************************** Datenblock in Records unterteilen
E2E2   20 2B DE   JSR $DE2B     Pufferzeiger setzen
E2E5   A0 02      LDY #$02
E2E7   A9 00      LDA #$00
E2E9   91 94      STA ($94),Y   Puffer löschen
E2EB   C8         INY
E2EC   D0 FB      BNE $E2E9
E2EE   20 04 E3   JSR $E304     Zeiger auf nächsten Record setzen
E2F1   95 C1      STA $C1,X
E2F3   A8         TAY
E2F4   A9 FF      LDA #$FF
E2F6   91 94      STA ($94),Y   $FF als erstes Zeichen des Records
E2F8   20 04 E3   JSR $E304     Zeiger auf nächsten Record setzen
E2FB   90 F4      BCC $E2F1     noch komplett in diesem Block ?
E2FD   D0 04      BNE $E303     Block voll ?
E2FF   A9 00      LDA #$00
E301   95 C1      STA $C1,X     Schreibzeiger auf Null
E303   60         RTS

****************************** Zeiger auf nächsten Record setzen
E304   A6 82      LDX $82       Kanalnummer
E306   B5 C1      LDA $C1,X     Schreibzeiger
E308   38         SEC
E309   F0 0D      BEQ $E318     gleich null ?
E30B   18         CLC
E30C   75 C7      ADC $C7,X     Recordlänge addieren
E30E   90 0B      BCC $E31B     kleiner als 256 ?
E310   D0 06      BNE $E318     gleich 256 ?
E312   A9 02      LDA #$02
E314   2C CC FE   BIT $FECC
E317   60         RTS

E318   69 01      ADC #$01      zwei addieren
E31A   38         SEC
E31B   60         RTS

****************************** Side-Sektoren erweitern
E31C   20 D3 D1   JSR $D1D3     Drivenummer holen
E31F   20 CB E1   JSR $E1CB     letzten Side-Sektor holen
E322   20 9C E2   JSR $E29C
E325   20 7B CF   JSR $CF7B
E328   A5 D6      LDA $D6
E32A   85 87      STA $87
E32C   A5 D5      LDA $D5       Side-Sektor Nummer
E32E   85 86      STA $86
E330   A9 00      LDA #$00
```

```
E332   85 88      STA $88
E334   A9 00      LDA #$00
E336   85 D4      STA $D4
E338   20 0E CE   JSR $CE0E    Side-Sektor Nummer und Zeiger berechnen
E33B   20 4D EF   JSR $EF4D    Zahl der freien Blöcke
E33E   A4 82      LDY $82      Kanalnummer
E340   B6 C7      LDX $C7,Y    Recordlänge
E342   CA         DEX
E343   8A         TXA
E344   18         CLC
E345   65 D7      ADC $D7      plus Zeiger in Datenblock
E347   90 0C      BCC $E355
E349   E6 D6      INC $D6
E34B   E6 D6      INC $D6      Zeiger auf Ende um zwei erhöhen (Track/Sektor)
E34D   D0 06      BNE $E355    kein Übertrag ?
E34F   E6 D5      INC $D5      Side-Sektor Nummer erhöhen
E351   A9 10      LDA #$10
E353   85 D6      STA $D6      Zeiger auf 16 setzen
E355   A5 87      LDA $87
E357   18         CLC
E358   69 02      ADC #$02
E35A   20 E9 DE   JSR $DEE9    Pufferzeiger für Side-Sektor setzen
E35D   A5 D5      LDA $D5      Side-Sektor Nummer
E35F   C9 06      CMP #$06
E361   90 05      BCC $E368    kleiner als 6 ?
E363   A9 52      LDA #$52
E365   20 C8 C1   JSR $C1C8    52, 'file too large'
E368   A5 D6      LDA $D6      Endezeiger
E36A   38         SEC
E36B   E5 87      SBC $87      minus letzter Endezeiger
E36D   B0 03      BCS $E372
E36F   E9 0F      SBC #$0F     minus 16
E371   18         CLC
E372   85 72      STA $72
E374   A5 D5      LDA $D5      Side-Sektor Nummer
E376   E5 86      SBC $86      minus letzte Side-Sektor Nummer
E378   85 73      STA $73      merken
E37A   A2 00      LDX #$00
E37C   86 70      STX $70      Summe für Brechnung löschen
E37E   86 71      STX $71
E380   AA         TAX
E381   20 51 DF   JSR $DF51    Blockzahl der Rel-Datei berechnen
E384   A5 71      LDA $71
E386   D0 07      BNE $E38F
E388   A6 70      LDX $70
E38A   CA         DEX
E38B   D0 02      BNE $E38F
E38D   E6 88      INC $88
E38F   CD 73 02   CMP $0273    Blockzahl der Rel-Datei
E392   90 09      BCC $E39D    größer als freie Blocks auf Diskette ?
E394   D0 CD      BNE $E363    52, 'file too large'
E396   AD 72 02   LDA $0272
E399   C5 70      CMP $70
E39B   90 C6      BCC $E363    52, 'file too large'
E39D   A9 01      LDA #$01
E39F   20 F6 D4   JSR $D4F6    Byte aus Puffer holen
```

```
E3A2  18        CLC
E3A3  69 01     ADC #$01      plus 1
E3A5  A6 82     LDX $82
E3A7  95 C1     STA $C1,X     als Schreibzeiger
E3A9  20 1E F1  JSR $F11E     freien Block in BAM suchen
E3AC  20 FD DD  JSR $DDFD     Track und Sektor in Puffer
E3AF  A5 88     LDA $88
E3B1  D0 15     BNE $E3C8     nur ein Block erforderlich ?
E3B3  20 5E DE  JSR $DE5E     Block schreiben
E3B6  20 1E CF  JSR $CF1E     Puffer wechseln
E3B9  20 D0 D6  JSR $D6D0     Parameter an Disk-Controller übergeben
E3BC  20 1E F1  JSR $F11E     freien Block in BAM suchen
E3BF  20 FD DD  JSR $DDFD     Track und Sektor in Puffer
E3C2  20 E2 E2  JSR $E2E2     Puffer löschen
E3C5  4C D4 E3  JMP $E3D4

E3C8  20 1E CF  JSR $CF1E     Puffer wechseln
E3CB  20 D0 D6  JSR $D6D0     Parameter an Disk-Controller übergeben
E3CE  20 E2 E2  JSR $E2E2     Puffer löschen
E3D1  20 19 DE  JSR $DE19     Nullbyte und Endezeiger in Puffer
E3D4  20 5E DE  JSR $DE5E     Block schreiben
E3D7  20 0C DE  JSR $DE0C     Track und Sektor holen
E3DA  A5 80     LDA $80       Track
E3DC  48        PHA
E3DD  A5 81     LDA $81       und Sektor
E3DF  48        PHA           merken
E3E0  20 3E DE  JSR $DE3E     Track und Sektor vom Disk-Controller holen
E3E3  A5 81     LDA $81       Sektor
E3E5  48        PHA
E3E6  A5 80     LDA $80       und Track merken
E3E8  48        PHA
E3E9  20 45 DF  JSR $DF45     Pufferzeiger für Side-Sektor setzen
E3EC  AA        TAX
E3ED  D0 0A     BNE $E3F9     Zeiger ungleich null ?
E3EF  20 4E E4  JSR $E44E     Side-Sektor schreiben, nächsten anlegen
E3F2  A9 10     LDA #$10
E3F4  20 E9 DE  JSR $DEE9     Pufferzeiger auf 16
E3F7  E6 86     INC $86       Side-Sektor Nummer erhöhen
E3F9  68        PLA
E3FA  20 8D DD  JSR $DD8D     Track in Side-Sektor
E3FD  68        PLA
E3FE  20 8D DD  JSR $DD8D     Sektor in Side-Sektor
E401  68        PLA
E402  85 81     STA $81       Sektor
E404  68        PLA
E405  85 80     STA $80       und Track zurückholen
E407  F0 0F     BEQ $E418     kein weiterer Block ?
E409  A5 86     LDA $86       Side-Sektor Nummer
E40B  C5 D5     CMP $D5       geändert ?
E40D  D0 A7     BNE $E3B6     ja
E40F  20 45 DF  JSR $DF45     Pufferzeiger in Side-Sektor setzen
E412  C5 D6     CMP $D6       Endezeiger
E414  90 A0     BCC $E3B6     kleiner ?
E416  F0 B0     BEQ $E3C8     gleich
E418  20 45 DF  JSR $DF45     Pufferzeiger in Side-Sektor setzen
E41B  48        PHA
```

```
E41C   A9 00      LDA #$00
E41E   20 DC DE   JSR $DEDC     Pufferzeiger auf null
E421   A9 00      LDA #$00
E423   A8         TAY
E424   91 94      STA ($94),Y   Null als Tracknummer
E426   C8         INY
E427   68         PLA           Endezeiger
E428   38         SEC
E429   E9 01      SBC #$01      minus eins
E42B   91 94      STA ($94),Y   als Sektor
E42D   20 6C DE   JSR $DE6C     Block schreiben
E430   20 99 D5   JSR $D599     und prüfen
E433   20 F4 EE   JSR $EEF4     BAM updaten
E436   20 0E CE   JSR $CE0E     Zeiger für Rel-Datei updaten
E439   20 1E CF   JSR $CF1E     Puffer wechseln
E43C   20 F8 DE   JSR $DEF8     richtiger Side-Sektor ?
E43F   70 03      BVS $E444     nein
E441   4C 75 E2   JMP $E275

E444   A9 80      LDA #$80
E446   20 97 DD   JSR $DD97     Bit 7 setzen
E449   A9 50      LDA #$50
E44B   20 C8 C1   JSR $C1C8     50, 'record not present'

****************************** Side-Sektor schreiben und neuen anlegen
E44E   20 1E F1   JSR $F11E     freien Block in BAM suchen
E451   20 1E CF   JSR $CF1E     Puffer wechseln
E454   20 F1 DD   JSR $DDF1     Block schreiben
E457   20 93 DF   JSR $DF93     Puffernummer holen
E45A   48         PHA
E45B   20 C1 DE   JSR $DEC1     Puffer löschen
E45E   A6 82      LDX $82       Kanalnummer
E460   B5 CD      LDA $CD,X     Puffernummer
E462   A8         TAY
E463   68         PLA
E464   AA         TAX
E465   A9 10      LDA #$10      16 Bytes des Side-Sektors
E467   20 A5 DE   JSR $DEA5     in Puffer kopieren
E46A   A9 00      LDA #$00
E46C   20 DC DE   JSR $DEDC     Pufferzeiger auf null, alter Side-Sektor
E46F   A0 02      LDY #$02
E471   B1 94      LDA ($94),Y   Side-Sektor Nummer
E473   48         PHA
E474   A9 00      LDA #$00
E476   20 C8 D4   JSR $D4C8     Pufferzeiger auf null, neuer Side-Sektor
E479   68         PLA
E47A   18         CLC
E47B   69 01      ADC #$01      Side-Sektor Nummer erhöhen
E47D   91 94      STA ($94),Y   und in Puffer
E47F   0A         ASL  A        mal 2
E480   69 04      ADC #$04      plus 4
E482   85 89      STA $89       gleich Zeiger auf Track/Sektor
E484   A8         TAY
E485   38         SEC
E486   E9 02      SBC #$02      minus 2
E488   85 8A      STA $8A       gleich Zeiger auf alten Side-Sektor
```

```
E48A   A5 80      LDA $80       Track
E48C   85 87      STA $87
E48E   91 94      STA ($94),Y   in Puffer
E490   C8         INY
E491   A5 81      LDA $81       Sektor
E493   85 88      STA $88
E495   91 94      STA ($94),Y   in Puffer
E497   A0 00      LDY #$00
E499   98         TYA
E49A   91 94      STA ($94),Y   Null in Puffer
E49C   C8         INY
E49D   A9 11      LDA #$11      17
E49F   91 94      STA ($94),Y   Anzahl der Bytes im Block
E4A1   A9 10      LDA #$10      16
E4A3   20 C8 D4   JSR $D4C8     Pufferzeiger auf 16
E4A6   20 50 DE   JSR $DE50     Block schreiben
E4A9   20 99 D5   JSR $D599     und prüfen
E4AC   A6 82      LDX $82       Kanalnummer
E4AE   B5 CD      LDA $CD,X     Puffernummer des Side-Sektors
E4B0   48         PHA
E4B1   20 9E DF   JSR $DF9E     Puffernummer holen
E4B4   A6 82      LDX $82       Kanalnummer
E4B6   95 CD      STA $CD,X     in Tabelle schreiben
E4B8   68         PLA
E4B9   AE 57 02   LDX $0257     Kanalnummer + 7
E4BC   95 A7      STA $A7,X     in Tabelle
E4BE   A9 00      LDA #$00
E4C0   20 C8 D4   JSR $D4C8     Pufferzeiger auf null
E4C3   A0 00      LDY #$00
E4C5   A5 80      LDA $80       Track
E4C7   91 94      STA ($94),Y   in Puffer
E4C9   C8         INY
E4CA   A5 81      LDA $81       Sektor
E4CC   91 94      STA ($94),Y   in Puffer
E4CE   4C DE E4   JMP $E4DE

E4D1   20 93 DF   JSR $DF93     Puffernummer holen
E4D4   A6 82      LDX $82       Kanalnummer
E4D6   20 1B DF   JSR $DF1B     Block lesen
E4D9   A9 00      LDA #$00
E4DB   20 C8 D4   JSR $D4C8     Pufferzeiger auf Null
E4DE   C6 8A      DEC $8A
E4E0   C6 8A      DEC $8A       Zähler für Side-Sektor Blocks
E4E2   A4 89      LDY $89
E4E4   A5 87      LDA $87       Tracknummer
E4E6   91 94      STA ($94),Y   in Puffer
E4E8   C8         INY
E4E9   A5 88      LDA $88       Sektornummer
E4EB   91 94      STA ($94),Y   in Puffer
E4ED   20 5E DE   JSR $DE5E     Block schreiben
E4F0   20 99 D5   JSR $D599     und prüfen
E4F3   A4 8A      LDY $8A       Zähler für Side-Sektor Blocks
E4F5   C0 03      CPY #$03
E4F7   B0 D8      BCS $E4D1     größer oder gleich 3 ?
E4F9   4C 1E CF   JMP $CF1E     Puffer wechseln
```

```
**************************      Tabelle der Fehlermeldungen
E4FC 00                         00
E4FD A0 4F CB                   ' oK'
E500 20 21 22 23 24 27          Fehlernummern der 'read error'
E506 D2 45 41 44                'Read'
E50A 89                         Zeiger auf 'error'
E50B 52                         52
E50C 83                         Zeiger auf 'file'
E50D 20 54 4F 4F 20 4C 41 52 47 C5   ' too largE'
E517 50                         50
E518 8B 06                      Zeiger auf 'record ' und 'not '
E51A 20 50 52 45 53 45 4E D4    ' presenT'
E522 51                         51
E523 CF 56 45 52 46 4C 4F 57 20 'Overflow in'
E52E 8B                         Zeiger auf 'record'
E52F 25 28                      Fehlernummern der 'write error'
E531 8A 89                      Zeiger auf 'write' und ' error'
E533 26                         26
E534 8A                         Zeiger auf 'write'
E535 20 50 52 4F 54 45 43 54 20 4F CE  ' protect oN'
E540 29                         29
E541 88                         Zeiger auf 'disk'
E542 20 49 85                   ' id'
E545 85                         Zeiger auf ' mismatch'
E546 30 31 32 33 34             Fehlernummern für 'syntax error'
E54B D3 59 4E 54 41 58          'Syntax'
E551 89                         Zeiger auf ' error'
E552 60                         60
E553 8A 03 84                   Zeiger auf 'write', 'file' und 'open'
E556 63                         63
E557 83                         Zeiger auf 'file'
E558 20 45 58 49 53 54 D3       ' existS'
E55F 64                         64
E560 83                         Zeiger auf 'file'
E561 20 54 59 50 45             ' type'
E566 85                         Zeiger auf 'mismatch'
E567 65                         65
E568 CE 4F 20 42 4C 4F 43 CB    'No blocK'
E570 66 67                      Fehlernummern für 'illegal track or sector'
E572 C9 4C 4C 45 47 41 4C 20    'Illegal '
E57A 54 52 41 43 4B 20 4F 52    'track or'
E582 20 53 45 43 54 4F D2       ' sectoR'
E589 61                         61
E58A 83 06 84                   Zeiger auf 'file', 'not' und 'open'
E58D 39 62                      Fehlernummern für 'file not found'
E590 83 06 87                   Zeiger auf 'file', 'not' und 'found'
E593 01                         01
E594 83                         Zeiger auf 'file'
E594 53 20 53 43 52 41 54 43 48 45 C4  's scratcheD'
E59F 70                         70
E5A0 CE 4F 20 43 48 41 4E 4E 45 CC  'No channeL'
E5AA 71                         71
E5AB C4 49 52                   'Dir'
E5AE 89                         Zeiger auf 'error'
E5AF 72                         72
E5B0 88                         Zeiger auf 'disk'
```

```
E5B1 20 46 55 4C CC             ' fulL'
E5B6 73                         73
E5B7 C3 42 4D 20 44 4F 53 20    'Cbm dos '
E5BF 56 32 2E 36 20 31 35 34 B1 'v2.6 1541'
E5C4 74                         74
E5C5 C4 52 49 56 45             'Drive'
E5CA 06                         Zeiger auf 'not'
E5CB 20 52 45 41 44 D9          ' readY'
E5D5 09
E5D6 C5 52 52 4F D2             'ErroR'
E5DB 0A
E5DC D7 52 49 54 C5             'WritE'
E5E1 03
E5E2 C6 49 4C C5                'FilE'
E5E6 04
E6E7 CF 50 45 CE                'OpeN'
E5EB 05
E5EC CD 49 53 4D 41 54 43 C8    'MismatcH'
E5F4 06
E5F5 CE 4F D4                   'NoT'
E5F8 07
E5F9 C6 4F 55 4E C4             'FounD'
E5FE 08
E5FF C4 49 53 CB                'DisK'
E603 0B
E604 D2 45 43 4F 52 C4          'RecorD'

******************************  Fehlernummer und -Meldung bereitstellen
E60A   48         PHA           Fehlerkode merken
E60B   86 F9      STX $F9       Drivenummer
E60D   8A         TXA
E60E   0A         ASL  A        mal 2
E60F   AA         TAX           als Zeiger
E610   B5 06      LDA $06,X
E612   85 80      STA $80       Track
E614   B5 07      LDA $07,X
E616   85 81      STA $81       und Sektornummer holen
E618   68         PLA           Fehlerkode zurückholen
E619   29 0F      AND #$0F      Bits 0 bis 3 isolieren
E61B   F0 08      BEQ $E625     null, dann 24, 'read error'
E61D   C9 0F      CMP #$0F      15 ?
E61F   D0 06      BNE $E627
E621   A9 74      LDA #$74
E623   D0 08      BNE $E62D     74, 'drive not ready'
E625   A9 06      LDA #$06      6
E627   09 20      ORA #$20      $20 addieren
E629   AA         TAX
E62A   CA         DEX
E62B   CA         DEX           zwei abziehen
E62C   8A         TXA
E62D   48         PHA           Fehlernummer merken
E62E   AD 2A 02   LDA $022A     Nummer des Diskettenkommandos
E631   C9 00      CMP #$00      OPEN oder VALIDATE ?
E633   D0 0F      BNE $E644     nein
E635   A9 FF      LDA #$FF
E637   8D 2A 02   STA $022A
```

```
E63A   68         PLA           Fehlernummer zurückholen
E63B   20 C7 E6   JSR $E6C7     Fehlermeldung generieren
E63E   20 42 D0   JSR $D042     BAM laden
E641   4C 48 E6   JMP $E648     Fehlermeldung setzen

E644   68         PLA
E645   20 C7 E6   JSR $E6C7     Fehlermeldung setzen
E648   20 BD C1   JSR $C1BD     Eingabepuffer löschen
E64B   A9 00      LDA #$00
E64D   8D F9 02   STA $02F9     Fehlerflag löschen
E650   20 2C C1   JSR $C12C     LED ausschalten
E653   20 DA D4   JSR $D4DA     Kanäle 17 und 18 schließen
E656   A9 00      LDA #$00
E658   85 A3      STA $A3       Zeiger Eingabepuffer auf Null
E65A   A2 45      LDX #$45
E65C   9A         TXS           Stackpointer initialisieren
E65D   A5 84      LDA $84       Sekundäradresse
E65F   29 0F      AND #$0F
E661   85 83      STA $83
E663   C9 0F      CMP #$0F      15 ?
E665   F0 31      BEQ $E698     ja, Kommandokanal
E667   78         SEI
E668   A5 79      LDA $79       LISTEN aktiv ?
E66A   D0 1C      BNE $E688     ja
E66C   A5 7A      LDA $7A       TALK aktiv ?
E66E   D0 10      BNE $E680     ja
E670   A6 83      LDX $83       Kanalnummer
E672   BD 2B 02   LDA $022B,X   Kanal zu dieser Sekundäradresse offen ?
E675   C9 FF      CMP #$FF
E677   F0 1F      BEQ $E698     nein
E679   29 0F      AND #$0F
E67B   85 82      STA $82       Kanalnummer
E67D   4C 8E E6   JMP $E68E

******************************  TALK
E680   20 EB D0   JSR $D0EB     Kanal zum Lesen öffnen
E683   20 4E EA   JSR $EA4E     Byte annehmen
E686   D0 06      BNE $E68E

******************************  LISTEN
E688   20 07 D1   JSR $D107     Kanal zum Schreiben öffnen
E68B   20 4E EA   JSR $EA4E     Byte annehmen
E68E   20 25 D1   JSR $D125     Dateityp prüfen
E691   C9 04      CMP #$04      Dateityp REL ?
E693   B0 03      BCS $E698     ja
E695   20 27 D2   JSR $D227     Kanal schließen
E698   4C E7 EB   JMP $EBE7

******************************  Hexzahl in Dezimalzahl wandlen (2 Bytes)
E69B   AA         TAX
E69C   A9 00      LDA #$00
E69E   F8         SED
E69F   E0 00      CPX #$00
E6A1   F0 07      BEQ $E6AA     Hexzahl in BCD-Zahl wandlen
E6A3   18         CLC
E6A4   69 01      ADC #$01
```

```
E6A6   CA          DEX
E6A7   4C 9F E6    JMP $E69F
E6AA   D8          CLD

****************************  BCD-Zahl in zwei Bytes zerlegen
E6AB   AA          TAX
E6AC   4A          LSR  A
E6AD   4A          LSR  A         Hi-Nibble nach unten verschieben
E6AE   4A          LSR  A
E6AF   4A          LSR  A
E6B0   20 B4 E6    JSR $E6B4      nach ASCII wandeln
E6B3   8A          TXA
E6B4   29 0F       AND #$0F       oberste 4 Bit löschen
E6B6   09 30       ORA #$30       '0' addieren
E6B8   91 A5       STA ($A5),Y    in Puffer schreiben
E6BA   C8          INY            Pufferzeiger erhöhen
E6BB   60          RTS

****************************  'ok' in Puffer schreiben
E6BC   20 23 C1    JSR $C123      Fehlerflags löschen
E6BF   A9 00       LDA #$00       Fehlernummer 0
E6C1   A0 00       LDY #$00
E6C3   84 80       STY $80        Track 0
E6C5   84 81       STY $81        Sektor 0

****************************  Fehlermeldung in Puffer (Nummer im Akku)
E6C7   A0 00       LDY #$00       Pufferzeiger
E6C9   A2 D5       LDX #$D5
E6CB   86 A5       STX $A5        Zeiger $A5/$A6 auf $2D5
E6CD   A2 02       LDX #$02
E6CF   86 A6       STX $A6
E6D1   20 AB E6    JSR $E6AB      Fehlernummer nach ASCII und in Puffer
E6D4   A9 2C       LDA #$2C       ',' Komma
E6D6   91 A5       STA ($A5),Y    in Puffer schreiben
E6D8   C8          INY            Pufferzeiger erhöhen
E6D9   AD D5 02    LDA $02D5      erste Ziffer des Diskettenstatus
E6DC   8D 43 02    STA $0243      ins Ausgaberegister
E6DF   8A          TXA            Fehlernummer in Akku
E6E0   20 06 E7    JSR $E706      Fehlermeldung in Puffer
E6E3   A9 2C       LDA #$2C       ',' Komma
E6E5   91 A5       STA ($A5),Y    in Puffer schreiben
E6E7   C8          INY            und Pufferzeiger erhöhen
E6E8   A5 80       LDA $80        Tracknummer
E6EA   20 9B E6    JSR $E69B      nach ASCII und in Puffer
E6ED   A9 2C       LDA #$2C       ',' Komma
E6EF   91 A5       STA ($A5),Y    in Puffer schreiben
E6F1   C8          INY            Pufferzeiger erhöhen
E6F2   A5 81       LDA $81        Sektor
E6F4   20 9B E6    JSR $E69B      nach ASCII wandeln und in Puffer
E6F7   88          DEY
E6F8   98          TYA
E6F9   18          CLC
E6FA   69 D5       ADC #$D5
E6FC   8D 49 02    STA $0249      Endezeiger
E6FF   E6 A5       INC $A5
E701   A9 88       LDA #$88       READ-Flag setzen
```

```
E703   85 F7      STA $F7
E705   60         RTS

****************************  Text der Fehlermeldung in Puffer schreiben
E706   AA         TAX           Fehlercode nach X
E707   A5 86      LDA $86
E709   48         PHA           Zeiger $86/$87 retten
E70A   A5 87      LDA $87
E70C   48         PHA
E70D   A9 FC      LDA #$FC
E70F   85 86      STA $86       Zeiger $86/$87 auf $E4FC
E711   A9 E4      LDA #$E4      Beginn der Fehlermeldungen
E713   85 87      STA $87
E715   8A         TXA           Fehlernummer in Akku
E716   A2 00      LDX #$00
E718   C1 86      CMP ($86,X)   mit Fehlernummer in Tabelle vergleichen
E71A   F0 21      BEQ $E73D
E71C   48         PHA
E71D   20 75 E7   JSR $E775     Bit 7 ins Carry und löschen
E720   90 05      BCC $E727     nicht gesetzt ?
E722   20 75 E7   JSR $E775     Bit 7 ins Carry
E725   90 FB      BCC $E722     warten auf Zeichen mit gesetztem Bit 7
E727   A5 87      LDA $87
E729   C9 E6      CMP #$E6
E72B   90 08      BCC $E735     $E60A, auf Ende der Tabelle prüfen
E72D   D0 0A      BNE $E739
E72F   A9 0A      LDA #$0A
E731   C5 86      CMP $86
E733   90 04      BCC $E739
E735   68         PLA
E736   4C 18 E7   JMP $E718     nein, weitermachen
E739   68         PLA
E73A   4C 4D E7   JMP $E74D     fertig

E73D   20 67 E7   JSR $E767     ein Zeichen holen, Bit 7 ins Carry
E740   90 FB      BCC $E73D     warten auf Zeichen mit Bit 7 gesetzt
E742   20 54 E7   JSR $E754     und in Puffer schreiben
E745   20 67 E7   JSR $E767     nächstes Zeichen holen
E748   90 F8      BCC $E742     warten auf Zeichen mit gesetztem Bit 7
E74A   20 54 E7   JSR $E754     Zeichen in Puffer
E74D   68         PLA
E74E   85 87      STA $87
E750   68         PLA           Zeiger $86/$87 zurückholen
E751   85 86      STA $86
E753   60         RTS

****************************  Zeichen holen und in Puffer
E754   C9 20      CMP #$20      ' ' Leerzeichen
E756   B0 0B      BCS $E763     größer, dann in Puffer schreiben
E758   AA         TAX           Kode merken
E759   A9 20      LDA #$20      Leerzeichen
E75B   91 A5      STA ($A5),Y   in Puffer schreiben
E75D   C8         INY           Pufferzeiger erhöhen
E75E   8A         TXA           Kode in Akku
E75F   20 06 E7   JSR $E706     zugehörigen Text ausgeben
E762   60         RTS
```

```
E763   91 A5      STA ($A5),Y   Zeichen in Puffer schreiben
E765   C8         INY           und Zeiger erhöhen
E766   60         RTS

****************************** ein Zeichen der Fehlermeldung holen
E767   E6 86      INC $86
E769   D0 02      BNE $E76D     Zeiger erhöhen
E76B   E6 87      INC $87
E76D   A1 86      LDA ($86,X)   Zeichen holen
E76F   0A         ASL  A        Bit 7 ins Carry
E770   A1 86      LDA ($86,X)   Zeichen holen
E772   29 7F      AND #$7F      Bit 7 löschen
E774   60         RTS

****************************** Zeiger erhöhen
E775   20 6D E7   JSR $E76D     Bit 7 ins Carry
E778   E6 86      INC $86
E77A   D0 02      BNE $E77E     Zeiger erhöhen
E77C   E6 87      INC $87
E77E   60         RTS

******************************
E77F   60         RTS

****************************** prüft auf AUTO-Start
E780   AD 00 18   LDA $1800     IEC-Port lesen
E783   AA         TAX
E784   29 04      AND #$04      'CLOCK IN'-Bit isolieren
E786   F0 F7      BEQ $E77F     nicht gesetzt, dann fertig
E788   8A         TXA
E789   29 01      AND #$01      'DATA IN'-Bit isolieren
E78B   F0 F2      BEQ $E77F     nicht gesetzt, dann ferig
E78D   58         CLI
E78E   AD 00 18   LDA $1800     IEC-Port laden
E791   29 05      AND #$05      'DATA IN' und 'CLOCK IN' testen
E793   D0 F9      BNE $E78E     warten bis beide gesetzt sind
E795   EE 78 02   INC $0278     ein Dateinamen
E798   EE 74 02   INC $0274     ein Zeichen in der Eingabezeile
E79B   A9 2A      LDA #$2A      '*' als Filename
E79D   8D 00 02   STA $0200     in Puffer schreiben
E7A0   4C A8 E7   JMP $E7A8

****************************** '&' - Befehl
E7A3   A9 8D      LDA #$8D
E7A5   20 68 C2   JSR $C268     Befehlszeile bis zum Ende prüfen
E7A8   20 58 F2   JSR $F258     (RTS)
E7AB   AD 78 02   LDA $0278     Zahl der Dateinamen
E7AE   48         PHA           merken
E7AF   A9 01      LDA #$01
E7B1   8D 78 02   STA $0278     ein Dateiname
E7B4   A9 FF      LDA #$FF
E7B6   85 86      STA $86
E7B8   20 4F C4   JSR $C44F     Datei suchen
E7BB   AD 80 02   LDA $0280
E7BE   D0 05      BNE $E7C5     gefunden ?
E7C0   A9 39      LDA #$39
```

```
E7C2   20 C8 C1   JSR $C1C8     39, 'file not found'
E7C5   68         PLA
E7C6   8D 78 02   STA $0278     Zahl der Dateinamen zurückholen
E7C9   AD 80 02   LDA $0280
E7CC   85 80      STA $80       Track
E7CE   AD 85 02   LDA $0285
E7D1   85 81      STA $81       und Sektor
E7D3   A9 03      LDA #$03      Dateityp 'USR'
E7D5   20 77 D4   JSR $D477     Puffer belegen, ersten Block lesen
E7D8   A9 00      LDA #$00
E7DA   85 87      STA $87       Prüfsumme löschen
E7DC   20 39 E8   JSR $E839     Byte aus Datei holen
E7DF   85 88      STA $88       als Startadresse lo merken
E7E1   20 4B E8   JSR $E84B     Prüfsumme bilden
E7E4   20 39 E8   JSR $E839     Byte aus Datei holen
E7E7   85 89      STA $89       als Startadresse hi
E7E9   20 4B E8   JSR $E84B     Prüfsumme bilden
E7EC   A5 86      LDA $86
E7EE   F0 0A      BEQ $E7FA
E7F0   A5 88      LDA $88
E7F2   48         PHA           Programmstartadresse merken
E7F3   A5 89      LDA $89
E7F5   48         PHA
E7F6   A9 00      LDA #$00
E7F8   85 86      STA $86
E7FA   20 39 E8   JSR $E839     Byte aus Datei holen
E7FD   85 8A      STA $8A       als Zähler merken
E7FF   20 4B E8   JSR $E84B     Prüfsumme bilden
E802   20 39 E8   JSR $E839     Byte aus Datei holen
E805   A0 00      LDY #$00
E807   91 88      STA ($88),Y   als Programmbytes abspeichern
E809   20 4B E8   JSR $E84B     Prüfsumme bilden
E80C   A5 88      LDA $88
E80E   18         CLC
E80F   69 01      ADC #$01
E811   85 88      STA $88       Zeiger $88/$89 erhöhen
E813   90 02      BCC $E817
E815   E6 89      INC $89
E817   C6 8A      DEC $8A       Zähler erniedrigen
E819   D0 E7      BNE $E802
E81B   20 35 CA   JSR $CA35     nächstes Byte holen
E81E   A5 85      LDA $85       Datenbyte
E820   C5 87      CMP $87       gleich Prüfsumme ?
E822   F0 08      BEQ $E82C     ja
E824   20 3E DE   JSR $DE3E     Parameter an Disk-Controller übergeben
E827   A9 50      LDA #$50
E829   20 45 E6   JSR $E645     50, 'record not present'
E82C   A5 F8      LDA $F8       Ende ?
E82E   D0 A8      BNE $E7D8     nein, nächster Datenblock
E830   68         PLA
E831   85 89      STA $89
E833   68         PLA           Programmstartadresse zurückholen
E834   85 88      STA $88
E836   6C 88 00   JMP ($0088)   und Programm ausführen
E839   20 35 CA   JSR $CA35     Byte aus Datei holen
E83C   A5 F8      LDA $F8       Ende ?
```

```
E83E   D0 08      BNE $E848     nein
E840   20 3E DE   JSR $DE3E     Parameter an Disk-Controller übergeben
E843   A9 51      LDA #$51
E845   20 45 E6   JSR $E645     51, 'overflow in record'
E848   A5 85      LDA $85       Datenbyte
E84A   60         RTS

******************************  Prüfsumme bilden
E84B   18         CLC
E84C   65 87      ADC $87
E84E   69 00      ADC #$00
E850   85 87      STA $87
E852   60         RTS

******************************  IRQ-Routine für seriellen Bus
E853   AD 01 18   LDA $1801     Port A lesen, IRQ-Flag löschen
E856   A9 01      LDA #$01
E858   85 7C      STA $7C       Flag für 'ATN empfangen' setzen
E85A   60         RTS

******************************  Bedienung des seriellen Bus
E85B   78         SEI
E85C   A9 00      LDA #$00
E85E   85 7C      STA $7C       Flag für 'ATN empfangen' löschen
E860   85 79      STA $79       Flag für LISTEN löschen
E862   85 7A      STA $7A       Flag für TALK löschen
E864   A2 45      LDX #$45
E866   9A         TXS           Stackpointer initialisieren
E867   A9 80      LDA #$80
E869   85 F8      STA $F8       Endeflag löschen
E86B   85 7D      STA $7D       EOI-Flag löschen
E86D   20 B7 E9   JSR $E9B7     CLOCK OUT lo
E870   20 A5 E9   JSR $E9A5     DATA OUT, Bit '0', hi
E873   AD 00 18   LDA $1800
E876   09 10      ORA #$10      Datenleitungen auf Eingabe schalten
E878   8D 00 18   STA $1800
E87B   AD 00 18   LDA $1800     IEC-Port lesen
E87E   10 57      BPL $E8D7     EOI ?
E880   29 04      AND #$04      CLOCK IN ?
E882   D0 F7      BNE $E87B     nein
E884   20 C9 E9   JSR $E9C9     Byte vom Bus holen
E887   C9 3F      CMP #$3F      Unlisten ?
E889   D0 06      BNE $E891     nein
E88B   A9 00      LDA #$00
E88D   85 79      STA $79       Flag für LISTEN rücksetzen
E88F   F0 71      BEQ $E902
E891   C9 5F      CMP #$5F      Untalk ?
E893   D0 06      BNE $E89B     nein
E895   A9 00      LDA #$00
E897   85 7A      STA $7A       Flag für TALK rücksetzen
E899   F0 67      BEQ $E902
E89B   C5 78      CMP $78       TALK-Adresse ?
E89D   D0 0A      BNE $E8A9     nein
E89F   A9 01      LDA #$01
E8A1   85 7A      STA $7A       Flag für TALK setzen
E8A3   A9 00      LDA #$00
```

```
E8A5   85 79      STA $79       Flag für LISTEN rücksetzen
E8A7   F0 29      BEQ $E8D2
E8A9   C5 77      CMP $77       LISTEN-Adresse ?
E8AB   D0 0A      BNE $E8B7     nein
E8AD   A9 01      LDA #$01
E8AF   85 79      STA $79       Flag für LISTEN setzen
E8B1   A9 00      LDA #$00
E8B3   85 7A      STA $7A       Flag für TALK rücksetzen
E8B5   F0 1B      BEQ $E8D2
E8B7   AA         TAX
E8B8   29 60      AND #$60
E8BA   C9 60      CMP #$60      Bit 5 und 6 gesetzt ?
E8BC   D0 3F      BNE $E8FD     nein
E8BE   8A         TXA
E8BF   85 84      STA $84       Byte ist Sekundäradresse
E8C1   29 0F      AND #$0F
E8C3   85 83      STA $83       Kanalnummer
E8C5   A5 84      LDA $84
E8C7   29 F0      AND #$F0
E8C9   C9 E0      CMP #$E0      CLOSE ?
E8CB   D0 35      BNE $E902
E8CD   58         CLI
E8CE   20 C0 DA   JSR $DAC0     CLOSE-Routine
E8D1   78         SEI
E8D2   2C 00 18   BIT $1800
E8D5   30 AD      BMI $E884
E8D7   A9 00      LDA #$00
E8D9   85 7D      STA $7D       EOI setzen
E8DB   AD 00 18   LDA $1800     IEC-Port
E8DE   29 EF      AND #$EF      Datenleitungen auf Ausgabe schalten
E8E0   8D 00 18   STA $1800
E8E3   A5 79      LDA $79       LISTEN aktiv ?
E8E5   F0 06      BEQ $E8ED     nein
E8E7   20 2E EA   JSR $EA2E     Daten empfangen
E8EA   4C E7 EB   JMP $EBE7     zur Warteschleife

E8ED   A5 7A      LDA $7A       TALK aktiv ?
E8EF   F0 09      BEQ $E8FA     nein
E8F1   20 9C E9   JSR $E99C     DATA OUT, Bit '1', lo
E8F4   20 AE E9   JSR $E9AE     CLOCK OUT hi
E8F7   20 09 E9   JSR $E909     Daten senden
E8FA   4C 4E EA   JMP $EA4E     zur Warteschleife

E8FD   A9 10      LDA #$10      weder TALK noch LISTEN, Byte ignorieren
E8FF   8D 00 18   STA $1800     Datenleitungen auf Eingabe schalten
E902   2C 00 18   BIT $1800
E905   10 D0      BPL $E8D7
E907   30 F9      BMI $E902     Handshake abwarten

****************************** Daten senden
E909   78         SEI
E90A   20 EB D0   JSR $D0EB     Kanal zum Lesen öffnen
E90D   B0 06      BCS $E915     Kanal aktiv ?
E90F   A6 82      LDX $82       Kanalnummer
E911   B5 F2      LDA $F2,X     READ-Flag gesetzt ?
E913   30 01      BMI $E916     ja
```

```
E915  60          RTS
E916  20 59 EA    JSR $EA59      auf EOI prüfen
E919  20 C0 E9    JSR $E9C0      IEC-Port lesen
E91C  29 01       AND #$01       Datenbit isolieren
E91E  08          PHP            und merken
E91F  20 B7 E9    JSR $E9B7      CLOCK OUT lo
E922  28          PLP
E923  F0 12       BEQ $E937
E925  20 59 EA    JSR $EA59      auf EOI prüfen
E928  20 C0 E9    JSR $E9C0      IEC-Port lesen
E92B  29 01       AND #$01       Datenbit isolieren
E92D  D0 F6       BNE $E925
E92F  A6 82       LDX $82        Kanalnummer
E931  B5 F2       LDA $F2,X
E933  29 08       AND #$08
E935  D0 14       BNE $E94B
E937  20 59 EA    JSR $EA59      auf EOI prüfen
E93A  20 C0 E9    JSR $E9C0      IEC-Port lesen
E93D  29 01       AND #$01       Datenbit isolieren
E93F  D0 F6       BNE $E937
E941  20 59 EA    JSR $EA59      auf EOI prüfen
E944  20 C0 E9    JSR $E9C0      IEC-Port lesen
E947  29 01       AND #$01       Datenbit isolieren
E949  F0 F6       BEQ $E941
E94B  20 AE E9    JSR $E9AE      CLOCK OUT hi
E94E  20 59 EA    JSR $EA59      auf EOI prüfen
E951  20 C0 E9    JSR $E9C0      IEC-Port lesen
E954  29 01       AND #$01       Datenbit isolieren
E956  D0 F3       BNE $E94B
E958  A9 08       LDA #$08       Zähler auf 8 Bits für serielle Übertragung
E95A  85 98       STA $98
E95C  20 C0 E9    JSR $E9C0      IEC-Port lesen
E95F  29 01       AND #$01       Datenbit isolieren
E961  D0 36       BNE $E999
E963  A6 82       LDX $82
E965  BD 3E 02    LDA $023E,X
E968  6A          ROR  A         unterstes Bit ins Carry
E969  9D 3E 02    STA $023E,X
E96C  B0 05       BCS $E973      Bit gesetzt
E96E  20 A5 E9    JSR $E9A5      DATA OUT, Bit '0' ausgeben
E971  D0 03       BNE $E976      unbedingter Sprung
E973  20 9C E9    JSR $E99C      DATA OUT, Bit '1' ausgeben
E976  20 B7 E9    JSR $E9B7      CLOCK OUT setzen
E979  A5 23       LDA $23
E97B  D0 03       BNE $E980
E97D  20 F3 FE    JSR $FEF3      Verzögerung für seriellen Bus
E980  20 FB FE    JSR $FEFB      DATA OUT und CLOCK OUT setzen
E983  C6 98       DEC $98        schon alle Bits ausgegeben ?
E985  D0 D5       BNE $E95C      nein
E987  20 59 EA    JSR $EA59      auf EOI prüfen
E98A  20 C0 E9    JSR $E9C0      IEC-Port lesen
E98D  29 01       AND #$01       Datenbit isolieren
E98F  F0 F6       BEQ $E987
E991  58          CLI
E992  20 AA D3    JSR $D3AA      nächstes Datenbyte holen
E995  78          SEI
```

```
E996   4C 0F E9   JMP $E90F      und ausgeben

E999   4C 4E EA   JMP $EA4E      zur Warteschleife

****************************** DATA OUT lo
E99C   AD 00 18   LDA $1800
E99F   29 FD      AND #$FD       Bit 'l' ausgeben
E9A1   8D 00 18   STA $1800
E9A4   60         RTS

****************************** DATA OUT hi
E9A5   AD 00 18   LDA $1800
E9A8   09 02      ORA #$02       Bit '0' ausgeben
E9AA   8D 00 18   STA $1800
E9AD   60         RTS

****************************** CLOCK OUT hi
E9AE   AD 00 18   LDA $1800
E9B1   09 08      ORA #$08       Bit 3 setzen
E9B3   8D 00 18   STA $1800
E9B6   60         RTS

****************************** CLOCK OUT lo
E9B7   AD 00 18   LDA $1800
E9BA   29 F7      AND #$F7       Bit 3 löschen
E9BC   8D 00 18   STA $1800
E9BF   60         RTS

****************************** IEC-Port lesen
E9C0   AD 00 18   LDA $1800      Port lesen
E9C3   CD 00 18   CMP $1800      konstanten Wert abwarten
E9C6   D0 F8      BNE $E9C0
E9C8   60         RTS

******************************
E9C9   A9 08      LDA #$08
E9CB   85 98      STA $98        Bitzähler serielle Ausgabe
E9CD   20 59 EA   JSR $EA59      auf EOI prüfen
E9D0   20 C0 E9   JSR $E9C0      IEC-Port lesen
E9D3   29 04      AND #$04       CLOCK IN ?
E9D5   D0 F6      BNE $E9CD      nein, warten
E9D7   20 9C E9   JSR $E99C      DATA OUT, Bit 'l'
E9DA   A9 01      LDA #$01
E9DC   8D 05 18   STA $1805      Timer setzen
E9DF   20 59 EA   JSR $EA59      auf EOI prüfen
E9E2   AD 0D 18   LDA $180D
E9E5   29 40      AND #$40       Timer abgelaufen ?
E9E7   D0 09      BNE $E9F2      ja, EOI
E9E9   20 C0 E9   JSR $E9C0      IEC-Port lesen
E9EC   29 04      AND #$04       CLOCK IN ?
E9EE   F0 EF      BEQ $E9DF      nein, warten
E9F0   D0 19      BNE $EA0B

E9F2   20 A5 E9   JSR $E9A5      DATA OUT Bit '0', hi
E9F5   A2 0A      LDX #$0A       10
E9F7   CA         DEX            Verzögerungsschleife, ca. 50 Mikrosekunden
```

```
E9F8   D0 FD      BNE $E9F7
E9FA   20 9C E9   JSR $E99C     DATA OUT, Bit '1', lo
E9FD   20 59 EA   JSR $EA59     auf EOI prüfen
EA00   20 C0 E9   JSR $E9C0     IEC-Port lesen
EA03   29 04      AND #$04      CLOCK IN ?
EA05   F0 F6      BEQ $E9FD     nein, warten
EA07   A9 00      LDA #$00
EA09   85 F8      STA $F8       EOI-Flag setzen
EA0B   AD 00 18   LDA $1800     IEC-Port
EA0E   49 01      EOR #$01      Datenbit invertieren
EA10   4A         LSR  A
EA11   29 02      AND #$02
EA13   D0 F6      BNE $EA0B     CLOCK IN ?
EA15   EA         NOP
EA16   EA         NOP
EA17   EA         NOP
EA18   66 85      ROR $85       nächstes Bit bereitstellen
EA1A   20 59 EA   JSR $EA59     auf EOI prüfen
EA1D   20 C0 E9   JSR $E9C0     IEC-Port lesen
EA20   29 04      AND #$04      CLOCK IN ?
EA22   F0 F6      BEQ $EA1A     nein
EA24   C6 98      DEC $98       Bitzähler erniedrigen
EA26   D0 E3      BNE $EA0B     alle Bits schon ausgegeben ?
EA28   20 A5 E9   JSR $E9A5     DATA OUT Bit '0', hi
EA2B   A5 85      LDA $85       Datenbyte wieder laden
EA2D   60         RTS

******************************  Datenannahme vom seriellen Bus
EA2E   78         SEI
EA2F   20 07 D1   JSR $D107     Kanal zum Schreiben öffnen
EA32   B0 05      BCS $EA39     Kanal nicht aktiv ?
EA34   B5 F2      LDA $F2,X     WRITE-Flag
EA36   6A         ROR  A
EA37   B0 0B      BCS $EA44     nicht gesetzt ?
EA39   A5 84      LDA $84       Sekundäradresse
EA3B   29 F0      AND #$F0
EA3D   C9 F0      CMP #$F0      OPEN-Befehl ?
EA3F   F0 03      BEQ $EA44     ja
EA41   4C 4E EA   JMP $EA4E     zur Warteschleife

EA44   20 C9 E9   JSR $E9C9     Datenbyte vom Bus holen
EA47   58         CLI
EA48   20 B7 CF   JSR $CFB7     und in Puffer schreiben
EA4B   4C 2E EA   JMP $EA2E     zum Schleifenanfang

EA4E   A9 00      LDA #$00
EA50   8D 00 18   STA $1800     IEC-Port rücksetzen
EA53   4C E7 EB   JMP $EBE7     zur Warteschleife

EA56   4C 5B E8   JMP $E85B     zur Hauptschleife serieller Bus

******************************
EA59   A5 7D      LDA $7D       EOI empfangen ?
EA5B   F0 06      BEQ $EA63     ja
EA5D   AD 00 18   LDA $1800     IEC-Port
EA60   10 09      BPL $EA6B
```

```
EA62   60          RTS

EA63   AD 00 18    LDA $1800      IEC-Port
EA66   10 FA       BPL $EA62
EA68   4C 5B E8    JMP $E85B      zur Hauptschleife serieller Bus

EA6B   4C D7 E8    JMP $E8D7      EOI setzen, seriellen Bus bedienen

****************************** LED-Blinken bei Hardwaredefekten, Selbsttest
EA6E   A2 00       LDX #$00       1 mal blinken, Zeropage
EA70   2C          .BYTE $2C
EA71   A6 6F       LDX $6F        X+1 mal blinken für RAM/ROM-Fehler
EA73   9A          TXS
EA74   BA          TSX
EA75   A9 08       LDA #$08       LED-Bit im Port auswählen
EA77   0D 00 1C    ORA $1C00
EA7A   4C EA FE    JMP $FEEA      LED einschalten, zurück nach $EA7D
EA7D   98          TYA
EA7E   18          CLC
EA7F   69 01       ADC #$01
EA81   D0 FC       BNE $EA7F
EA83   88          DEY
EA84   D0 F8       BNE $EA7E
EA86   AD 00 1C    LDA $1C00
EA89   29 F7       AND #$F7       LED ausschalten
EA8B   8D 00 1C    STA $1C00
EA8E   98          TYA
EA8F   18          CLC
EA90   69 01       ADC #$01
EA92   D0 FC       BNE $EA90      Verzögerungsschleife
EA94   88          DEY
EA95   D0 F8       BNE $EA8F
EA97   CA          DEX
EA98   10 DB       BPL $EA75
EA9A   E0 FC       CPX #$FC
EA9C   D0 F0       BNE $EA8E      Verzögerung abwarten
EA9E   F0 D4       BEQ $EA74      LED wieder einschalten

****************************** RESET-Routine
EAA0   78          SEI
EAA1   D8          CLD
EAA2   A2 FF       LDX #$FF
EAA4   8E 03 18    STX $1803      Port A auf Ausgabe
EAA7   E8          INX
EAA8   A0 00       LDY #$00
EAAA   A2 00       LDX #$00
EAAC   8A          TXA
EAAD   95 00       STA $00,X      Zeropage löschen
EAAF   E8          INX
EAB0   D0 FA       BNE $EAAC
EAB2   8A          TXA
EAB3   D5 00       CMP $00,X      ist Byte gelöscht ?
EAB5   D0 B7       BNE $EA6E      nein, dann zur Fehleranzeige (blinken)
EAB7   F6 00       INC $00,X
EAB9   C8          INY
EABA   D0 FB       BNE $EAB7
```

```
EABC   D5 00      CMP $00,X
EABE   D0 AE      BNE $EA6E      Fehler
EAC0   94 00      STY $00,X
EAC2   B5 00      LDA $00,X
EAC4   D0 A8      BNE $EA6E      Fehler
EAC6   E8         INX
EAC7   D0 E9      BNE $EAB2
EAC9   E6 6F      INC $6F
EACB   86 76      STX $76
EACD   A9 00      LDA #$00
EACF   85 75      STA $75
EAD1   A8         TAY
EAD2   A2 20      LDX #$20       32 Pages testen
EAD4   18         CLC
EAD5   C6 76      DEC $76
EAD7   71 75      ADC ($75),Y
EAD9   C8         INY
EADA   D0 FB      BNE $EAD7
EADC   CA         DEX
EADD   D0 F6      BNE $EAD5      ROM Testen
EADF   69 00      ADC #$00
EAE1   AA         TAX
EAE2   C5 76      CMP $76
EAE4   D0 39      BNE $EB1F      ROM-Fehler
EAE6   E0 C0      CPX #$C0
EAE8   D0 DF      BNE $EAC9
EAEA   A9 01      LDA #$01
EAEC   85 76      STA $76
EAEE   E6 6F      INC $6F
EAF0   A2 07      LDX #$07       RAM testen, beginnend bei Page 7
EAF2   98         TYA
EAF3   18         CLC
EAF4   65 76      ADC $76
EAF6   91 75      STA ($75),Y
EAF8   C8         INY
EAF9   D0 F7      BNE $EAF2
EAFB   E6 76      INC $76
EAFD   CA         DEX
EAFE   D0 F2      BNE $EAF2
EB00   A2 07      LDX #$07
EB02   C6 76      DEC $76
EB04   88         DEY
EB05   98         TYA
EB06   18         CLC
EB07   65 76      ADC $76
EB09   D1 75      CMP ($75),Y
EB0B   D0 12      BNE $EB1F      RAM-Fehler
EB0D   49 FF      EOR #$FF
EB0F   91 75      STA ($75),Y
EB11   51 75      EOR ($75),Y
EB13   91 75      STA ($75),Y
EB15   D0 08      BNE $EB1F      RAM-Fehler
EB17   98         TYA
EB18   D0 EA      BNE $EB04
EB1A   CA         DEX
EB1B   D0 E5      BNE $EB02      weiter testen
```

```
EB1D  F0 03     BEQ $EB22     ok
EB1F  4C 71 EA  JMP $EA71     zur Fehleranzeige

EB22  A2 45     LDX #$45
EB24  9A        TXS           Stackpointer initialisieren
EB25  AD 00 1C  LDA $1C00
EB28  29 F7     AND #$F7      LED ausschalten
EB2A  8D 00 1C  STA $1C00
EB2D  A9 01     LDA #$01
EB2F  8D 0C 18  STA $180C     CA1 (ATN IN) auf positive Flanke triggern
EB32  A9 82     LDA #$82
EB34  8D 0D 18  STA $180D     Interrupt durch ATN IN ermöglichen
EB37  8D 0E 18  STA $180E              "
EB3A  AD 00 18  LDA $1800     Port B lesen
EB3D  29 60     AND #$60      Bit 5 und 6 isolieren (Gerätenummer)
EB3F  0A        ASL  A
EB40  2A        ROL  A
EB41  2A        ROL  A        nach Bitposition 0 und 1 schieben
EB42  2A        ROL  A
EB43  09 48     ORA #$48      Offset von 8 plus $40 für Talk addieren
EB45  85 78     STA $78       Gerätenummer für TALK (senden)
EB47  49 60     EOR #$60      Bit 6 löschen, Bit 5 setzen
EB49  85 77     STA $77       Gerätenummer plus $20 für LISTEN (empfangen)
EB4B  A2 00     LDX #$00
EB4D  A0 00     LDY #$00
EB4F  A9 00     LDA #$00
EB51  95 99     STA $99,X     Low-Byte der Pufferadressen
EB53  E8        INX
EB54  B9 E0 FE  LDA $FEE0,Y   High-Byte der Adresse aus Tabelle
EB57  95 99     STA $99,X     speichern
EB59  E8        INX
EB5A  C8        INY
EB5B  C0 05     CPY #$05      5 Pufferspeicher
EB5D  D0 F0     BNE $EB4F
EB5F  A9 00     LDA #$00
EB61  95 99     STA $99,X
EB63  E8        INX           Zeiger $A3/$A4 auf $200, Eingabepuffer
EB64  A9 02     LDA #$02
EB66  95 99     STA $99,X
EB68  E8        INX
EB69  A9 D5     LDA #$D5
EB6B  95 99     STA $99,X
EB6D  E8        INX           Zeiger $A5/$A6 auf $2D5, Puffer Fehlermeldung
EB6E  A9 02     LDA #$02
EB70  95 99     STA $99,X
EB72  A9 FF     LDA #$FF
EB74  A2 12     LDX #$12
EB76  9D 2B 02  STA $022B,X   Kanaltabelle mit $FF füllen ('nicht belegt')
EB79  CA        DEX
EB7A  10 FA     BPL $EB76
EB7C  A2 05     LDX #$05
EB7E  95 A7     STA $A7,X     Puffertabellen löschen
EB80  95 AE     STA $AE,X
EB82  95 CD     STA $CD,X     Side-Sektor Tabelle löschen
EB84  CA        DEX
EB85  10 F7     BPL $EB7E
```

```
EB87   A9 05      LDA #$05       Puffer 5
EB89   85 AB      STA $AB        Kanal 4 zuordnen
EB8B   A9 06      LDA #$06       Puffer 6
EB8D   85 AC      STA $AC        Kanal 5 zuordnen
EB8F   A9 FF      LDA #$FF
EB91   85 AD      STA $AD
EB93   85 B4      STA $B4
EB95   A9 05      LDA #$05
EB97   8D 3B 02   STA $023B      Kanal 5 WRITE-Flag gelöscht
EB9A   A9 84      LDA #$84
EB9C   8D 3A 02   STA $023A      Kanal 4 WRITE-Flag gesetzt
EB9F   A9 0F      LDA #$0F       Kanalbelegungsregister initialisieren
EBA1   8D 56 02   STA $0256      Bit '1' gleich Kanal frei
EBA4   A9 01      LDA #$01
EBA6   85 F6      STA $F6        WRITE-Flag
EBA8   A9 88      LDA #$88
EBAA   85 F7      STA $F7        READ-Flag
EBAC   A9 E0      LDA #$E0       5 Puffer frei
EBAE   8D 4F 02   STA $024F      Pufferbelegungsregister initialisieren
EBB1   A9 FF      LDA #$FF       $24F/$250, 16 Bit, '1' gleich Puffer belegt
EBB3   8D 50 02   STA $0250
EBB6   A9 01      LDA #$01
EBB8   85 1C      STA $1C        Flags für Write Protect
EBBA   85 1D      STA $1D
EBBC   20 63 CB   JSR $CB63      Vektor für U0 setzen
EBBF   20 FA CE   JSR $CEFA      Kanaltabelle initialisieren
EBC2   20 59 F2   JSR $F259      Initialisierung für Disk-Controller
EBC5   A9 22      LDA #$22
EBC7   85 65      STA $65
EBC9   A9 EB      LDA #$EB       Zeiger $65/$66 auf $EB22
EBCB   85 66      STA $66
EBCD   A9 0A      LDA #$0A
EBCF   85 69      STA $69        Schrittweite 10 bei Sektorzuweisung
EBD1   A9 05      LDA #$05
EBD3   85 6A      STA $6A        5 Leseversuche
EBD5   A9 73      LDA #$73       Einschaltmeldung bereitstellen
EBD7   20 C1 E6   JSR $E6C1      73, 'cbm dos v2.6 1541'
EBDA   A9 1A      LDA #$1A       DATA OUT, CLOCK OUT und ATN A auf Ausgang
EBDC   8D 02 18   STA $1802      Datenrichtung Port B
EBDF   A9 00      LDA #$00
EBE1   8D 00 18   STA $1800      Datenregister löschen
EBE4   20 80 E7   JSR $E780      prüft auf Auto-Start
EBE7   58         CLI
EBE8   AD 00 18   LDA $1800
EBEB   29 E5      AND #$E5       seriellen Port rücksetzen
EBED   8D 00 18   STA $1800
EBF0   AD 55 02   LDA $0255      Kommandoflag gesetzt ?
EBF3   F0 0A      BEQ $EBFF      nein
EBF5   A9 00      LDA #$00
EBF7   8D 55 02   STA $0255      Kommandoflag rücksetzen
EBFA   85 67      STA $67
EBFC   20 46 C1   JSR $C146      Befehl analysieren und ausführen

****************************** Warteschleife
EBFF   58         CLI
EC00   A5 7C      LDA $7C        ATN-Signal entdeckt ?
```

```
EC02   F0 03      BEQ $EC07      nein
EC04   4C 5B E8   JMP $E85B      zur IEC-Routine
EC07   58         CLI
EC08   A9 0E      LDA #$0E       14
EC0A   85 72      STA $72        als Sekundäradresse
EC0C   A9 00      LDA #$00
EC0E   85 6F      STA $6F        Job-Zähler
EC10   85 70      STA $70
EC12   A6 72      LDX $72
EC14   BD 2B 02   LDA $022B,X    Sekundäradresse
EC17   C9 FF      CMP #$FF       Kanal zugeordnet ?
EC19   F0 10      BEQ $EC2B      nein
EC1B   29 3F      AND #$3F
EC1D   85 82      STA $82        Kanalnummer
EC1F   20 93 DF   JSR $DF93      Puffernummer holen
EC22   AA         TAX
EC23   BD 5B 02   LDA $025B,X   Drivenummer
EC26   29 01      AND #$01
EC28   AA         TAX
EC29   F6 6F      INC $6F,X      Jobzähler erhöhen
EC2B   C6 72      DEC $72        Lo-Adresse
EC2D   10 E3      BPL $EC12      weiter suchen
EC2F   A0 04      LDY #$04       Pufferzähler
EC31   B9 00 00   LDA $0000,Y    Disk-Controller in Aktion ?
EC34   10 05      BPL $EC3B      nein
EC36   29 01      AND #$01       Drivenummer isolieren
EC38   AA         TAX
EC39   F6 6F      INC $6F,X      Jobzähler erhöhen
EC3B   88         DEY
EC3C   10 F3      BPL $EC31      nächster Puffer
EC3E   78         SEI
EC3F   AD 00 1C   LDA $1C00
EC42   29 F7      AND #$F7       LED-Bit löschen
EC44   48         PHA
EC45   A5 7F      LDA $7F        Drivenummer
EC47   85 86      STA $86
EC49   A9 00      LDA #$00
EC4B   85 7F      STA $7F        Drive 0
EC4D   A5 6F      LDA $6F        Job für Drive 0 ?
EC4F   F0 0B      BEQ $EC5C      nein
EC51   A5 1C      LDA $1C        Write Protect für Drive 0 ?
EC53   F0 03      BEQ $EC58      nein
EC55   20 13 D3   JSR $D313      alle Kanäle zu Drive 0 schließen
EC58   68         PLA
EC59   09 08      ORA #$08       LED-Bit setzen
EC5B   48         PHA
EC5C   E6 7F      INC $7F        Drivenummer erhöhen
EC5E   A5 70      LDA $70        Job für Drive 1 ?
EC60   F0 0B      BEQ $EC6D      nein
EC62   A5 1D      LDA $1D        Write Protect für Drive 1 ?
EC64   F0 03      BEQ $EC69      nein
EC66   20 13 D3   JSR $D313      alle Kanäle zu Drive 1 schließen
EC69   68         PLA
EC6A   09 00      ORA #$00
EC6C   48         PHA
EC6D   A5 86      LDA $86
```

```
EC6F   85 7F      STA $7F       Drivenummer zurückholen
EC71   68         PLA           Bit für LED
EC72   AE 6C 02   LDX $026C     Interruptzähler für LED-Blinken
EC75   F0 21      BEQ $EC98     auf null ?
EC77   AD 00 1C   LDA $1C00
EC7A   E0 80      CPX #$80
EC7C   D0 03      BNE $EC81
EC7E   4C 8B EC   JMP $EC8B
EC81   AE 05 18   LDX $1805     Timerinterrupt löschen
EC84   30 12      BMI $EC98
EC86   A2 A0      LDX #$A0
EC88   8E 05 18   STX $1805     Timer setzen
EC8B   CE 6C 02   DEC $026C     Zähler erniedrigen
EC8E   D0 08      BNE $EC98     noch nicht null ?
EC90   4D 6D 02   EOR $026D     Zustandsbit für LED umkehren
EC93   A2 10      LDX #$10
EC95   8E 6C 02   STX $026C     Zähler neu setzen
EC98   8D 00 1C   STA $1C00     LED ein/ausschalten
EC9B   4C FF EB   JMP $EBFF     zurück zur Warteschleife

****************************** LOAD "$"
EC9E   A9 00      LDA #$00
ECA0   85 83      STA $83       Sekundäradresse Null
ECA2   A9 01      LDA #$01
ECA4   20 E2 D1   JSR $D1E2     Kanal und Puffer suchen
ECA7   A9 00      LDA #$00
ECA9   20 C8 D4   JSR $D4C8     Pufferzeiger initialisieren
ECAC   A6 82      LDX $82       Kanalnummer
ECAE   A9 00      LDA #$00
ECB0   9D 44 02   STA $0244,X   Zeiger auf Ende gleich Null
ECB3   20 93 DF   JSR $DF93     Puffernummer holen
ECB6   AA         TAX
ECB7   A5 7F      LDA $7F       Drivenummer
ECB9   9D 5B 02   STA $025B,X   in Tabelle bringen
ECBC   A9 01      LDA #$01      1
ECBE   20 F1 CF   JSR $CFF1     in Puffer schreiben
ECC1   A9 04      LDA #$04      4, Startadresse $0401
ECC3   20 F1 CF   JSR $CFF1     in Puffer schreiben
ECC6   A9 01      LDA #$01      2 mal 1
ECC8   20 F1 CF   JSR $CFF1
ECCB   20 F1 CF   JSR $CFF1     als Linkadresse in Puffer schreiben
ECCE   AD 72 02   LDA $0272     Drivenummer
ECD1   20 F1 CF   JSR $CFF1     als Zeilennummer in Puffer schreiben
ECD4   A9 00      LDA #$00      Zeilennummer hi
ECD6   20 F1 CF   JSR $CFF1     in Puffer
ECD9   20 59 ED   JSR $ED59     Directoryeintrag in Puffer
ECDC   20 93 DF   JSR $DF93     Puffernummer holen
ECDF   0A         ASL  A
ECE0   AA         TAX
ECE1   D6 99      DEC $99,X     Pufferzeiger erniedrigen
ECE3   D6 99      DEC $99,X
ECE5   A9 00      LDA #$00
ECE7   20 F1 CF   JSR $CFF1     0 als Zeilenende in Puffer
ECEA   A9 01      LDA #$01
ECEC   20 F1 CF   JSR $CFF1     2 mal 1 als Linkadresse
ECEF   20 F1 CF   JSR $CFF1
```

```
ECF2   20 CE C6   JSR $C6CE     Directoryeintrag in Puffer
ECF5   90 2C      BCC $ED23     weiterer Eintrag ?
ECF7   AD 72 02   LDA $0272     Blockzahl lo
ECFA   20 F1 CF   JSR $CFF1     in Puffer
ECFD   AD 73 02   LDA $0273     Blockzahl hi
ED00   20 F1 CF   JSR $CFF1     in Puffer
ED03   20 59 ED   JSR $ED59     Directoryeintrag in Puffer
ED06   A9 00      LDA #$00
ED08   20 F1 CF   JSR $CFF1     Null als Endekennzeichen in Puffer
ED0B   D0 DD      BNE $ECEA     Puffer voll ? nein
ED0D   20 93 DF   JSR $DF93     Puffernummer holen
ED10   0A         ASL  A
ED11   AA         TAX
ED12   A9 00      LDA #$00
ED14   95 99      STA $99,X     Pufferzeiger auf Null
ED16   A9 88      LDA #$88      READ-Flag setzen
ED18   A4 82      LDY $82       Kanalnummer
ED1A   8D 54 02   STA $0254
ED1D   99 F2 00   STA $00F2,Y   Flag für Kanal
ED20   A5 85      LDA $85       Datenbyte
ED22   60         RTS

*******************************
ED23   AD 72 02   LDA $0272     Blockzahl lo
ED26   20 F1 CF   JSR $CFF1     in Puffer schreiben
ED29   AD 73 02   LDA $0273     Blockzahl hi
ED2C   20 F1 CF   JSR $CFF1     in Puffer
ED2F   20 59 ED   JSR $ED59     'Blocks free.' in Puffer
ED32   20 93 DF   JSR $DF93     Puffernummer holen
ED35   0A         ASL  A        mal 2
ED36   AA         TAX
ED37   D6 99      DEC $99,X
ED39   D6 99      DEC $99,X     Pufferzeiger minus 2
ED3B   A9 00      LDA #$00
ED3D   20 F1 CF   JSR $CFF1
ED40   20 F1 CF   JSR $CFF1     dreimal Null als Programmende
ED43   20 F1 CF   JSR $CFF1
ED46   20 93 DF   JSR $DF93     Puffernummer holen
ED49   0A         ASL  A        mal 2
ED4A   A8         TAY
ED4B   B9 99 00   LDA $0099,Y   Pufferzeiger
ED4E   A6 82      LDX $82
ED50   9D 44 02   STA $0244,X   als Endekennzeichen
ED53   DE 44 02   DEC $0244,X
ED56   4C 0D ED   JMP $ED0D

*******************************  Directoryzeile übertragen
ED59   A0 00      LDY #$00
ED5B   B9 B1 02   LDA $02B1,Y   Zeichen aus Puffer
ED5E   20 F1 CF   JSR $CFF1     in Ausgabepuffer schreiben
ED61   C8         INY
ED62   C0 1B      CPY #$1B      schon 27 Zeichen ?
ED64   D0 F5      BNE $ED5B
ED66   60         RTS

*******************************  Byte aus Puffer holen
```

```
ED67   20 37 D1   JSR $D137      Byte aus Puffer holen
ED6A   F0 01      BEQ $ED6D      Pufferzeiger null ?
ED6C   60         RTS

ED6D   85 85      STA $85        Datenbyte merken
ED6F   A4 82      LDY $82        Kanalnummer
ED71   B9 44 02   LDA $0244,Y    Endekennzeichen setzen
ED74   F0 08      BEQ $ED7E      Null (LOAD $) ?
ED76   A9 80      LDA #$80
ED78   99 F2 00   STA $00F2,Y    READ-Flag setzen
ED7B   A5 85      LDA $85        Datenbyte
ED7D   60         RTS

ED7E   48         PHA
ED7F   20 EA EC   JSR $ECEA      Directoryzeile im Puffer erzeugen
ED82   68         PLA
ED83   60         RTS

****************************** V-Befehl,   'Collect'
ED84   20 D1 C1   JSR $C1D1      Eingabezeile auf Laufwerknummer durchsuchen
ED87   20 42 D0   JSR $D042      BAM laden
ED8A   A9 40      LDA #$40
ED8C   8D F9 02   STA $02F9
ED8F   20 B7 EE   JSR $EEB7      neue BAM im Puffer erzeugen
ED92   A9 00      LDA #$00
ED94   8D 92 02   STA $0292
ED97   20 AC C5   JSR $C5AC      Directory laden, ersten Eintrag suchen
ED9A   D0 3D      BNE $EDD9      gefunden ?
ED9C   A9 00      LDA #$00
ED9E   85 81      STA $81        Sektor 0
EDA0   AD 85 FE   LDA $FE85      18
EDA3   85 80      STA $80        Track 18 für BAM
EDA5   20 E5 ED   JSR $EDE5      Directoryblöcke als belegt kennzeichnen
EDA8   A9 00      LDA #$00
EDAA   8D F9 02   STA $02F9
EDAD   20 FF EE   JSR $EEFF      BAM auf Diskette zurückschreiben
EDB0   4C 94 C1   JMP $C194      fertig, Diskstatus bereitstellen

******************************
EDB3   C8         INY
EDB4   B1 94      LDA ($94),Y    Track und
EDB6   48         PHA
EDB7   C8         INY
EDB8   B1 94      LDA ($94),Y    Sektor merken
EDBA   48         PHA
EDBB   A0 13      LDY #$13       Zeiger auf Side-Sektor-Block
EDBD   B1 94      LDA ($94),Y
EDBF   F0 0A      BEQ $EDCB      kein Folgeblock ?
EDC1   85 80      STA $80        Track
EDC3   C8         INY
EDC4   B1 94      LDA ($94),Y
EDC6   85 81      STA $81        und Sektor des ersten Side-Sektor-Blocks
EDC8   20 E5 ED   JSR $EDE5      Side-Sektor Blöcke als belegt kennzeichnen
EDCB   68         PLA
EDCC   85 81      STA $81
EDCE   68         PLA            Track und Sektor zurückholen
```

```
EDCF   85 80      STA $80
EDD1   20 E5 ED   JSR $EDE5     Blöcke der Datei als belegt kennzeichnen
EDD4   20 04 C6   JSR $C604     nächsten Eintrag im Directory lesen
EDD7   F0 C3      BEQ $ED9C     Ende des Directorys ?
EDD9   A0 00      LDY #$00
EDDB   B1 94      LDA ($94),Y   Filetyp
EDDD   30 D4      BMI $EDB3     Bit 7 gesetzt, File geschlossen ?
EDDF   20 B6 C8   JSR $C8B6     Dateityp auf Null und BAM schreiben
EDE2   4C D4 ED   JMP $EDD4

****************************  Dateiblöcke in BAM belegen
EDE5   20 5F D5   JSR $D55F     Track und Sektornummer prüfen
EDE8   20 90 EF   JSR $EF90     Block in BAM belegen
EDEB   20 75 D4   JSR $D475     nächsten Block lesen
EDEE   A9 00      LDA #$00
EDF0   20 C8 D4   JSR $D4C8     Pufferzeiger auf Null
EDF3   20 37 D1   JSR $D137     Byte aus Puffer holen
EDF6   85 80      STA $80       Track
EDF8   20 37 D1   JSR $D137     Byte aus Puffer holen
EDFB   85 81      STA $81       Sektor
EDFD   A5 80      LDA $80       folgt weiterer Block ?
EDFF   D0 03      BNE $EE04     ja
EE01   4C 27 D2   JMP $D227     Kanal schließen

EE04   20 90 EF   JSR $EF90     Block in BAM belegen
EE07   20 4D D4   JSR $D44D     nächsten Block lesen
EE0A   4C EE ED   JMP $EDEE     weiter

****************************  N-Befehl, 'Header'
EE0D   20 12 C3   JSR $C312     Drivenummer holen
EE10   A5 E2      LDA $E2       Drivenummer
EE12   10 05      BPL $EE19     nicht eindeutig ?
EE14   A9 33      LDA #$33
EE16   4C C8 C1   JMP $C1C8     33, 'syntax error'
EE19   29 01      AND #$01
EE1B   85 7F      STA $7F       Drivenummer
EE1D   20 00 C1   JSR $C100     LED einschalten
EE20   A5 7F      LDA $7F       Drivenummer
EE22   0A         ASL  A        mal 2
EE23   AA         TAX
EE24   AC 7B 02   LDY $027B     Kommaposition
EE27   CC 74 02   CPY $0274     mit Ende Name vergleichen
EE2A   F0 1A      BEQ $EE46     Formatieren ohne ID
EE2C   B9 00 02   LDA $0200,Y   erstes Zeichen der ID
EE2F   95 12      STA $12,X     speichern
EE31   B9 01 02   LDA $0201,Y   zweites Zeichen
EE34   95 13      STA $13,X
EE36   20 07 D3   JSR $D307     alle Kanäle schließen
EE39   A9 01      LDA #$01
EE3B   85 80      STA $80       Track 1
EE3D   20 C6 C8   JSR $C8C6     Diskette formatieren
EE40   20 05 F0   JSR $F005     Puffer löschen
EE43   4C 56 EE   JMP $EE56     weiter wie unten

EE46   20 42 D0   JSR $D042     BAM laden
EE49   A6 7F      LDX $7F       Drivenummer
```

```
EE4B   BD 01 01   LDA $0101,X
EE4E   CD D5 FE   CMP $FED5      'A', Kennzeichen für 1541-Format
EE51   F0 03      BEQ $EE56      ok
EE53   4C 72 D5   JMP $D572      73, 'cbm dos v2.6 1541'

EE56   20 B7 EE   JSR $EEB7      BAM erzeugen
EE59   A5 F9      LDA $F9        Puffernummer
EE5B   A8         TAY
EE5C   0A         ASL  A
EE5D   AA         TAX
EE5E   AD 88 FE   LDA $FE88      $90, Beginn Diskname
EE61   95 99      STA $99,X      Pufferzeiger auf Name
EE63   AE 7A 02   LDX $027A
EE66   A9 1B      LDA #$1B       27
EE68   20 6E C6   JSR $C66E      Filenamen in Puffer schreiben
EE6B   A0 12      LDY #$12       Position 18
EE6D   A6 7F      LDX $7F        Drivenummer
EE6F   AD D5 FE   LDA $FED5      'A', 1541-Format
EE72   9D 01 01   STA $0101,X
EE75   8A         TXA
EE76   0A         ASL  A         mal 2
EE77   AA         TAX
EE78   B5 12      LDA $12,X      ID, erstes Zeichen
EE7A   91 94      STA ($94),Y    in Puffer
EE7C   C8         INY
EE7D   B5 13      LDA $13,X      und zweites Zeichen
EE7F   91 94      STA ($94),Y    in Puffer
EE81   C8         INY
EE82   C8         INY
EE83   A9 32      LDA #$32       '2'
EE85   91 94      STA ($94),Y    in Puffer
EE87   C8         INY
EE88   AD D5 FE   LDA $FED5      'A'  1541-Format
EE8B   91 94      STA ($94),Y    in Puffer
EE8D   A0 02      LDY #$02
EE8F   91 6D      STA ($6D),Y    und an Position 2
EE91   AD 85 FE   LDA $FE85      18
EE94   85 80      STA $80        Tracknummer
EE96   20 93 EF   JSR $EF93      Block als belegt kennzeichnen
EE99   A9 01      LDA #$01       1
EE9B   85 81      STA $81        Sektornummer
EE9D   20 93 EF   JSR $EF93      Block als belegt kennzeichnen
EEA0   20 FF EE   JSR $EEFF      BAM schreiben
EEA3   20 05 F0   JSR $F005      Zeiger $6D/$6E auf Puffer, Puffer löschen
EEA6   A0 01      LDY #$01
EEA8   A9 FF      LDA #$FF
EEAA   91 6D      STA ($6D),Y    Folgetrack 0, $FF gleich Anzahl gültige Bytes
EEAC   20 64 D4   JSR $D464      Block schreiben
EEAF   C6 81      DEC $81        Sektornummer erniedrigen, 0
EEB1   20 60 D4   JSR $D460      Block lesen
EEB4   4C 94 C1   JMP $C194      Diskstatus bereit stellen

****************************** BAM erzeugen
EEB7   20 D1 F0   JSR $F0D1
EEBA   A0 00      LDY #$00
EEBC   A9 12      LDA #$12       18
```

```
EEBE   91 6D      STA ($6D),Y   Zeiger auf Directory-Track
EEC0   C8         INY
EEC1   98         TYA           1
EEC2   91 6D      STA ($6D),Y   Zeiger auf Directory-Sektor
EEC4   C8         INY
EEC5   C8         INY
EEC6   C8         INY
EEC7   A9 00      LDA #$00
EEC9   85 6F      STA $6F
EECB   85 70      STA $70       3 Bytes gleich 24 Bits für Sektoren
EECD   85 71      STA $71
EECF   98         TYA           Byteposition
EED0   4A         LSR  A
EED1   4A         LSR  A        durch 4 ergibt Tracknummer
EED2   20 4B F2   JSR $F24B     Anzahl Sektoren holen
EED5   91 6D      STA ($6D),Y   und in BAM
EED7   C8         INY
EED8   AA         TAX
EED9   38         SEC
EEDA   26 6F      ROL $6F
EEDC   26 70      ROL $70       Bitmuster erzeugen
EEDE   26 71      ROL $71
EEE0   CA         DEX
EEE1   D0 F6      BNE $EED9
EEE3   B5 6F      LDA $6F,X     3 Bytes
EEE5   91 6D      STA ($6D),Y   der BAM in Puffer
EEE7   C8         INY
EEE8   E8         INX
EEE9   E0 03      CPX #$03
EEEB   90 F6      BCC $EEE3
EEED   C0 90      CPY #$90      schon Position 144 ?
EEEF   90 D6      BCC $EEC7     nein, nächsten Track
EEF1   4C 75 D0   JMP $D075     Anzahl freie Blocks berechnen

****************************  BAM bei Bedarf schreiben
EEF4   20 93 DF   JSR $DF93     Puffernummer holen
EEF7   AA         TAX
EEF8   BD 5B 02   LDA $025B,X   Befehl für Disk-Controller
EEFB   29 01      AND #$01
EEFD   85 7F      STA $7F       Drivenummer isolieren
EEFF   A4 7F      LDY $7F
EF01   B9 51 02   LDA $0251,Y   BAM-Änderungsflag gesetzt ?
EF04   D0 01      BNE $EF07     ja
EF06   60         RTS

EF07   A9 00      LDA #$00
EF09   99 51 02   STA $0251,Y   BAM-Änderungsflag rücksetzen
EF0C   20 3A EF   JSR $EF3A     Pufferzeiger für BAM setzen
EF0F   A5 7F      LDA $7F       Drivenummer
EF11   0A         ASL  A        mal 2
EF12   48         PHA
EF13   20 A5 F0   JSR $F0A5     BAM-Eintrag überprüfen
EF16   68         PLA
EF17   18         CLC
EF18   69 01      ADC #$01      Tracknummer erhöhen
EF1A   20 A5 F0   JSR $F0A5     BAM-Eintrag überprüfen
```

```
EF1D   A5 80      LDA $80       Track
EF1F   48         PHA
EF20   A9 01      LDA #$01
EF22   85 80      STA $80       Track 1
EF24   0A         ASL  A
EF25   0A         ASL  A        mal 4
EF26   85 6D      STA $6D
EF28   20 20 F2   JSR $F220     überprüft BAM
EF2B   E6 80      INC $80       Tracknummer erhöhen
EF2D   A5 80      LDA $80
EF2F   CD D7 FE   CMP $FED7     und mit Maximalwert plus 1 = 36 vergleichen
EF32   90 F0      BCC $EF24     ok, nächster Track
EF34   68         PLA
EF35   85 80      STA $80       Tracknummer zurückholen
EF37   4C 8A D5   JMP $D58A     BAM auf Diskette schreiben

******************************  Pufferzeiger für BAM setzen
EF3A   20 0F F1   JSR $F10F     6 für Drive 0 holen
EF3D   AA         TAX
EF3E   20 DF F0   JSR $F0DF     Puffer belegen
EF41   A6 F9      LDX $F9       Puffernummer
EF43   BD E0 FE   LDA $FEE0,X   Pufferadresse, hi Byte
EF46   85 6E      STA $6E
EF48   A9 00      LDA #$00      lo Byte
EF4A   85 6D      STA $6D       Zeiger nach $6D/$6E
EF4C   60         RTS

******************************  Anzahl freie Blocks für Directory holen
EF4D   A6 7F      LDX $7F       Drivenummer
EF4F   BD FA 02   LDA $02FA,X   Anzahl Blocks lo
EF52   8D 72 02   STA $0272
EF55   BD FC 02   LDA $02FC,X   Anzahl Blocks hi
EF58   8D 73 02   STA $0273     in Puffer für Directory
EF5B   60         RTS

******************************  Block als frei kennzeichnen
EF5C   20 F1 EF   JSR $EFF1     Pufferzeiger setzen
EF5F   20 CF EF   JSR $EFCF     Bit für Sektor in BAM löschen
EF62   38         SEC
EF63   D0 22      BNE $EF87     Block bereits frei, dann fertig
EF65   B1 6D      LDA ($6D),Y   Bitmuster der BAM
EF67   1D E9 EF   ORA $EFE9,X   Bit X setzen, Kennzeichen für frei
EF6A   91 6D      STA ($6D),Y
EF6C   20 88 EF   JSR $EF88     Flag für BAM geändert setzen
EF6F   A4 6F      LDY $6F
EF71   18         CLC
EF72   B1 6D      LDA ($6D),Y
EF74   69 01      ADC #$01      Anzahl der freien Blocks pro Track erhöhen
EF76   91 6D      STA ($6D),Y
EF78   A5 80      LDA $80       Track
EF7A   CD 85 FE   CMP $FE85     gleich 18 ?
EF7D   F0 3B      BEQ $EFBA     dann übergehen
EF7F   FE FA 02   INC $02FA,X   Anzahl der freien Blocks der Diskette erhöhen
EF82   D0 03      BNE $EF87
EF84   FE FC 02   INC $02FC,X   Anzahl Blocks hi erhöhen
EF87   60         RTS
```

```
******************************  Flag für 'BAM geändert' setzen
EF88   A6 7F      LDX $7F       Drivenummer
EF8A   A9 01      LDA #$01
EF8C   9D 51 02   STA $0251,X   Flag gleich eins
EF8F   60         RTS

******************************  Block als belegt kennzeichnen
EF90   20 F1 EF   JSR $EFF1     Pufferzeiger setzen
EF93   20 CF EF   JSR $EFCF     Bit für Sektor in BAM löschen
EF96   F0 36      BEQ $EFCE     bereits belegt, dann fertig
EF98   B1 6D      LDA ($6D),Y
EF9A   5D E9 EF   EOR $EFE9,X   Bit des Blocks umkehren (löschen)
EF9D   91 6D      STA ($6D),Y
EF9F   20 88 EF   JSR $EF88     Flag für BAM geändert setzen
EFA2   A4 6F      LDY $6F
EFA4   B1 6D      LDA ($6D),Y
EFA6   38         SEC
EFA7   E9 01      SBC #$01      Anzahl der Blocks pro Track erniedrigen
EFA9   91 6D      STA ($6D),Y
EFAB   A5 80      LDA $80       Track
EFAD   CD 85 FE   CMP $FE85     18 ?
EFB0   F0 0B      BEQ $EFBD     Directorytrack aussparen
EFB2   BD FA 02   LDA $02FA,X   Anzahl freie Blocks lo
EFB5   D0 03      BNE $EFBA
EFB7   DE FC 02   DEC $02FC,X   Anzahl der freien Blocks erniedrigen
EFBA   DE FA 02   DEC $02FA,X
EFBD   BD FC 02   LDA $02FC,X   Anzahl freie Blocks hi
EFC0   D0 0C      BNE $EFCE     mehr als 255 Blocks frei ?
EFC2   BD FA 02   LDA $02FA,X   freie Blocks lo
EFC5   C9 03      CMP #$03
EFC7   B0 05      BCS $EFCE     weniger als 3 ?
EFC9   A9 72      LDA #$72
EFCB   20 C7 E6   JSR $E6C7     72, 'disk full'
EFCE   60         RTS

******************************  Bit für Sektor in BAM-Eintrag löschen
EFCF   20 11 F0   JSR $F011     sucht BAM-Feld für diesen Track
EFD2   98         TYA
EFD3   85 6F      STA $6F
EFD5   A5 81      LDA $81       Sektor
EFD7   4A         LSR  A
EFD8   4A         LSR  A        durch 8 teilen
EFD9   4A         LSR  A
EFDA   38         SEC
EFDB   65 6F      ADC $6F
EFDD   A8         TAY           Bytenummer in BAM-Eintrag
EFDE   A5 81      LDA $81       Sektornummer
EFE0   29 07      AND #$07
EFE2   AA         TAX           Bitnummer in BAM-Eintrag
EFE3   B1 6D      LDA ($6D),Y   Byte in BAM
EFE5   3D E9 EF   AND $EFE9,X   Bit für Sektor löschen entspricht belegt
EFE8   60         RTS

******************************  Zweierpotenzen
EFE9   01 02 04 08 10 20 40 80
```

```
*****************************  BAM nach Änderung schreiben
EFF1   A9 FF      LDA #$FF
EFF3   2C F9 02   BIT $02F9
EFF6   F0 0C      BEQ $F004
EFF8   10 0A      BPL $F004
EFFA   70 08      BVS $F004
EFFC   A9 00      LDA #$00
EFFE   8D F9 02   STA $02F9     Flag rücksetzen
F001   4C 8A D5   JMP $D58A     Block schreiben
F004   60         RTS
*****************************  BAM-Puffer löschen
F005   20 3A EF   JSR $EF3A     Zeiger $6D/$6E auf BAM-Puffer
F008   A0 00      LDY #$00
F00A   98         TYA
F00B   91 6D      STA ($6D),Y   BAM-Puffer löschen
F00D   C8         INY
F00E   D0 FB      BNE $F00B
F010   60         RTS

*****************************
F011   A5 6F      LDA $6F
F013   48         PHA
F014   A5 70      LDA $70
F016   48         PHA
F017   A6 7F      LDX $7F       Drivenummer
F019   B5 FF      LDA $FF,X
F01B   F0 05      BEQ $F022     Drive Null ?
F01D   A9 74      LDA #$74
F01F   20 48 E6   JSR $E648     'drive not ready'
F022   20 0F F1   JSR $F10F     Puffernummer für BAM holen
F025   85 6F      STA $6F
F027   8A         TXA
F028   0A         ASL  A
F029   85 70      STA $70
F02B   AA         TAX
F02C   A5 80      LDA $80       Track
F02E   DD 9D 02   CMP $029D,X
F031   F0 0B      BEQ $F03E
F033   E8         INX
F034   86 70      STX $70
F036   DD 9D 02   CMP $029D,X
F039   F0 03      BEQ $F03E
F03B   20 5B F0   JSR $F05B
F03E   A5 70      LDA $70
F040   A6 7F      LDX $7F       Drivenummer
F042   9D 9B 02   STA $029B,X
F045   0A         ASL  A
F046   0A         ASL  A        mal 4
F047   18         CLC
F048   69 A1      ADC #$A1
F04A   85 6D      STA $6D
F04C   A9 02      LDA #$02
F04E   69 00      ADC #$00
F050   85 6E      STA $6E
F052   A0 00      LDY #$00
F054   68         PLA
```

```
F055   85 70      STA $70
F057   68         PLA
F058   85 6F      STA $6F
F05A   60         RTS

*****************************
F05B   A6 6F      LDX $6F
F05D   20 DF F0   JSR $F0DF
F060   A5 7F      LDA $7F        Drivenummer
F062   AA         TAX
F063   0A         ASL  A
F064   1D 9B 02   ORA $029B,X
F067   49 01      EOR #$01
F069   29 03      AND #$03
F06B   85 70      STA $70
F06D   20 A5 F0   JSR $F0A5
F070   A5 F9      LDA $F9        Puffernummer
F072   0A         ASL  A
F073   AA         TAX
F074   A5 80      LDA $80        Track
F076   0A         ASL  A
F077   0A         ASL  A         mal 4
F078   95 99      STA $99,X      gleich Zeiger in BAM-Feld
F07A   A5 70      LDA $70
F07C   0A         ASL  A
F07D   0A         ASL  A
F07E   A8         TAY
F07F   A1 99      LDA ($99,X)
F081   99 A1 02   STA $02A1,Y
F084   A9 00      LDA #$00
F086   81 99      STA ($99,X)    Null in Puffer
F088   F6 99      INC $99,X      Pufferzeiger erhöhen
F08A   C8         INY
F08B   98         TYA
F08C   29 03      AND #$03
F08E   D0 EF      BNE $F07F
F090   A6 70      LDX $70
F092   A5 80      LDA $80        Track
F094   9D 9D 02   STA $029D,X
F097   AD F9 02   LDA $02F9
F09A   D0 03      BNE $F09F
F09C   4C 8A D5   JMP $D58A      Block schreiben

F09F   09 80      ORA #$80
F0A1   8D F9 02   STA $02F9
F0A4   60         RTS

F0A5   A8         TAY
F0A6   B9 9D 02   LDA $029D,Y
F0A9   F0 25      BEQ $F0D0
F0AB   48         PHA
F0AC   A9 00      LDA #$00
F0AE   99 9D 02   STA $029D,Y
F0B1   A5 F9      LDA $F9        Puffernummr
F0B3   0A         ASL  A         mal 2
F0B4   AA         TAX
```

```
F0B5   68         PLA
F0B6   0A         ASL  A
F0B7   0A         ASL  A
F0B8   95 99      STA $99,X
F0BA   98         TYA
F0BB   0A         ASL  A
F0BC   0A         ASL  A
F0BD   A8         TAY
F0BE   B9 A1 02   LDA $02A1,Y
F0C1   81 99      STA ($99,X)   in Puffer schreiben
F0C3   A9 00      LDA #$00
F0C5   99 A1 02   STA $02A1,Y
F0C8   F6 99      INC $99,X     Pufferzeiger erhöhen
F0CA   C8         INY
F0CB   98         TYA
F0CC   29 03      AND #$03
F0CE   D0 EE      BNE $F0BE
F0D0   60         RTS

F0D1   A5 7F      LDA $7F       Drivenummer
F0D3   0A         ASL  A
F0D4   AA         TAX
F0D5   A9 00      LDA #$00
F0D7   9D 9D 02   STA $029D,X
F0DA   E8         INX
F0DB   9D 9D 02   STA $029D,X
F0DE   60         RTS

F0DF   B5 A7      LDA $A7,X
F0E1   C9 FF      CMP #$FF
F0E3   D0 25      BNE $F10A
F0E5   8A         TXA
F0E6   48         PHA
F0E7   20 8E D2   JSR $D28E
F0EA   AA         TAX
F0EB   10 05      BPL $F0F2
F0ED   A9 70      LDA #$70
F0EF   20 C8 C1   JSR $C1C8     70, 'no channel'
F0F2   86 F9      STX $F9
F0F4   68         PLA
F0F5   A8         TAY
F0F6   8A         TXA
F0F7   09 80      ORA #$80
F0F9   99 A7 00   STA $00A7,Y
F0FC   0A         ASL  A
F0FD   AA         TAX
F0FE   AD 85 FE   LDA $FE85     18, Directorytrack
F101   95 06      STA $06,X     merken
F103   A9 00      LDA #$00      0
F105   95 07      STA $07,X     als Sektor
F107   4C 86 D5   JMP $D586     Block schreiben

F10A   29 0F      AND #$0F
F10C   85 F9      STA $F9       Puffernummer
F10E   60         RTS
```

```
****************************  Puffernummer für BAM holen
F10F   A9 06      LDA #$06
F111   A6 7F      LDX $7F       Drivenummer
F113   D0 03      BNE $F118
F115   18         CLC
F116   69 07      ADC #$07      gibt 13 für Drive 0
F118   60         RTS

****************************  Puffernummer für BAM nach X
F119   20 0F F1   JSR $F10F     Puffernummer holen
F11C   AA         TAX
F11D   60         RTS

****************************  freien Block in BAM suchen und belegen
F11E   20 3E DE   JSR $DE3E     Track und Sektornummer holen
F121   A9 03      LDA #$03
F123   85 6F      STA $6F       Zähler
F125   A9 01      LDA #$01
F127   0D F9 02   ORA $02F9
F12A   8D F9 02   STA $02F9
F12D   A5 6F      LDA $6F       Zähler merken
F12F   48         PHA
F130   20 11 F0   JSR $F011     BAM-Feld zu diesem Track suchen
F133   68         PLA
F134   85 6F      STA $6F       Zähler zurückholen
F136   B1 6D      LDA ($6D),Y   Anzahl der freien Bytes des Tracks
F138   D0 39      BNE $F173     noch Blocks frei ?
F13A   A5 80      LDA $80       Track
F13C   CD 85 FE   CMP $FE85     18, Directorytrack ?
F13F   F0 19      BEQ $F15A     ja, 'disk full'
F141   90 1C      BCC $F15F     kleiner, dann nächst niedrigerer Track
F143   E6 80      INC $80       Tracknummer erhöhen
F145   A5 80      LDA $80
F147   CD D7 FE   CMP $FED7     36, höchste Tracknummer plus eins
F14A   D0 E1      BNE $F12D     nein, auf diesem Track weitersuchen
F14C   AE 85 FE   LDX $FE85     18, Directorytrack
F14F   CA         DEX           erniedrigen
F150   86 80      STX $80       als Tracknummer merken
F152   A9 00      LDA #$00
F154   85 81      STA $81       mit Sektornummer null beginnen
F156   C6 6F      DEC $6F       Zähler erniedrigen
F158   D0 D3      BNE $F12D     noch nicht null, dann weitersuchen
F15A   A9 72      LDA #$72
F15C   20 C8 C1   JSR $C1C8     72, 'disk full'
F15F   C6 80      DEC $80       Tracknummer erniedrigen
F161   D0 CA      BNE $F12D     noch nicht null, in diesem Track weitersuchen
F163   AE 85 FE   LDX $FE85     18, Directorytrack
F166   E8         INX           erhöhen
F167   86 80      STX $80       als Tracknummer merken
F169   A9 00      LDA #$00
F16B   85 81      STA $81       mit Sektor null beginnen
F16D   C6 6F      DEC $6F       Zähler erniedrigen
F16F   D0 BC      BNE $F12D     noch nicht null, dann weiter suchen
F171   F0 E7      BEQ $F15A     sonst 'disk full'

F173   A5 81      LDA $81       Sektornummer
```

```
F175   18          CLC
F176   65 69       ADC $69        plus Schrittweite (10)
F178   85 81       STA $81        als neue Nummer
F17A   A5 80       LDA $80        Tracknummer
F17C   20 4B F2    JSR $F24B      maximale Sektornummer holen
F17F   8D 4E 02    STA $024E
F182   8D 4D 02    STA $024D      und merken
F185   C5 81       CMP $81        größer als gewählte Sektornummer ?
F187   B0 0C       BCS $F195      ja
F189   38          SEC            sonst
F18A   A5 81       LDA $81        Sektornummer
F18C   ED 4E 02    SBC $024E      minus maximale Sektornummer
F18F   85 81       STA $81        als neue Sektornummer merken
F191   F0 02       BEQ $F195      null ?
F193   C6 81       DEC $81        sonst Sektornummer um eins erniedrigen
F195   20 FA F1    JSR $F1FA      BAM prüfen, freien Sektor suchen
F198   F0 03       BEQ $F19D      nicht gefunden ?
F19A   4C 90 EF    JMP $EF90      Block in der BAM belegen
F19D   A9 00       LDA #$00
F19F   85 81       STA $81        Sektor Null
F1A1   20 FA F1    JSR $F1FA      freien Sektor ab Nummer 0 suchen
F1A4   D0 F4       BNE $F19A      gefunden ?
F1A6   4C F5 F1    JMP $F1F5      nein, 'dir error'

******************************    freien Sektor suchen und belegen
F1A9   A9 01       LDA #$01
F1AB   0D F9 02    ORA $02F9
F1AE   8D F9 02    STA $02F9
F1B1   A5 86       LDA $86
F1B3   48          PHA
F1B4   A9 01       LDA #$01       Trackzähler
F1B6   85 86       STA $86
F1B8   AD 85 FE    LDA $FE85      18, Directorytrack
F1BB   38          SEC
F1BC   E5 86       SBC $86        minus Zähler
F1BE   85 80       STA $80        als Tracknummer merken
F1C0   90 09       BCC $F1CB      Ergebnis kleiner gleich Null ?
F1C2   F0 07       BEQ $F1CB      dann oberhalb Directory versuchen
F1C4   20 11 F0    JSR $F011      BAM-Feld zu diesem Track suchen
F1C7   B1 6D       LDA ($6D),Y    Anzahl der freien Blocks in diesem Track
F1C9   D0 1B       BNE $F1E6      freie Blocks vorhanden
F1CB   AD 85 FE    LDA $FE85      18, Directorytrack
F1CE   18          CLC
F1CF   65 86       ADC $86        plus Zähler
F1D1   85 80       STA $80        als Tracknummer merken
F1D3   E6 86       INC $86        Zähler erhöhen
F1D5   CD D7 FE    CMP $FED7      36, maximale Tracknummer plus eins
F1D8   90 05       BCC $F1DF      kleiner, dann ok
F1DA   A9 67       LDA #$67
F1DC   20 45 E6    JSR $E645      67, 'illegal track or sector'
F1DF   20 11 F0    JSR $F011      BAM-Feld zu diesem Track suchen
F1E2   B1 6D       LDA ($6D),Y    Anzahl der freien Blocks in diesem Track
F1E4   F0 D2       BEQ $F1B8      kein Block mehr frei ?
F1E6   68          PLA
F1E7   85 86       STA $86
F1E9   A9 00       LDA #$00
```

```
F1EB   85 81      STA $81       Sektor 0
F1ED   20 FA F1   JSR $F1FA     freien Sektor suchen
F1F0   F0 03      BEQ $F1F5     nicht gefunden ?
F1F2   4C 90 EF   JMP $EF90     Block in BAM belegen

F1F5   A9 71      LDA #$71
F1F7   20 45 E6   JSR $E645     71, 'dir error'

****************************** freien Sektor auf aktuellem Track suchen
F1FA   20 11 F0   JSR $F011     BAM-Feld zu diesem Track suchen
F1FD   98         TYA           zeigt auf Anzahl der freien Blocks
F1FE   48         PHA
F1FF   20 20 F2   JSR $F220     BAM überprüfen
F202   A5 80      LDA $80       Track
F204   20 4B F2   JSR $F24B     maximale Sektornummer des Tracks holen
F207   8D 4E 02   STA $024E     merken
F20A   68         PLA
F20B   85 6F      STA $6F       Zeiger merken
F20D   A5 81      LDA $81       Sektor
F20F   CD 4E 02   CMP $024E     mit Maximalzahl vergleichen
F212   B0 09      BCS $F21D     größer oder gleich ?
F214   20 D5 EF   JSR $EFD5     Bitnummer des Sektors holen
F217   D0 06      BNE $F21F     Sektor frei ?
F219   E6 81      INC $81       Sektornummer erhöhen
F21B   D0 F0      BNE $F20D     und prüfen ob frei
F21D   A9 00      LDA #$00      kein Sektor frei
F21F   60         RTS

****************************** Anzahl freie Blocks in BAM überprüfen
F220   A5 6F      LDA $6F
F222   48         PHA
F223   A9 00      LDA #$00
F225   85 6F      STA $6F       Zähler auf null
F227   AC 86 FE   LDY $FE86     4, Anzahl Bytes pro Track in der BAM
F22A   88         DEY
F22B   A2 07      LDX #$07
F22D   B1 6D      LDA ($6D),Y
F22F   3D E9 EF   AND $EFE9,X   Bit isolieren
F232   F0 02      BEQ $F236
F234   E6 6F      INC $6F       bei freiem Sektor Zähler erhöhen
F236   CA         DEX
F237   10 F4      BPL $F22D
F239   88         DEY
F23A   D0 EF      BNE $F22B
F23C   B1 6D      LDA ($6D),Y   mit Anzahl auf Diskette vergleichen
F23E   C5 6F      CMP $6F
F240   D0 04      BNE $F246     ungleich, dann Fehler
F242   68         PLA
F243   85 6F      STA $6F
F245   60         RTS
F246   A9 71      LDA #$71
F248   20 45 E6   JSR $E645     71, 'dir error'

****************************** Anzahl Sektoren pro Track bestimmen
F24B   AE D6 FE   LDX $FED6     4 verschiedene Werte
F24E   DD D6 FE   CMP $FED6,X   Tracknummer
```

```
F251   CA          DEX
F252   B0 FA       BCS $F24E      noch größer ?
F254   BD D1 FE    LDA $FED1,X    Anzahl der Sektoren holen
F257   60          RTS

F258   60          RTS

****************************** Initialisierung für Disk Controller
F259   A9 6F       LDA #$6F       Bit 4 (Write Protect) und 7 (SYNC) Eingang
F25B   8D 02 1C    STA $1C02      Datenrichtungsregister Port B
F25E   29 F0       AND #$F0
F260   8D 00 1C    STA $1C00      Port B, Steuerport
F263   AD 0C 1C    LDA $1C0C      PCR, Kontrollregister
F266   29 FE       AND #$FE
F268   09 0E       ORA #$0E
F26A   09 E0       ORA #$E0
F26C   8D 0C 1C    STA $1C0C
F26F   A9 41       LDA #$41
F271   8D 0B 1C    STA $1C0B      Timer 1 free running, Port A Latch enable
F274   A9 00       LDA #$00
F276   8D 06 1C    STA $1C06      Timer 1 lo Latch
F279   A9 3A       LDA #$3A
F27B   8D 07 1C    STA $1C07      Timer 1 Hi Latch
F27E   8D 05 1C    STA $1C05      Timer 1 Hi
F281   A9 7F       LDA #$7F
F283   8D 0E 1C    STA $1C0E      IRQs löschen
F286   A9 C0       LDA #$C0
F288   8D 0D 1C    STA $1C0D
F28B   8D 0E 1C    STA $1C0E      IER, Interrupts erlauben
F28E   A9 FF       LDA #$FF
F290   85 3E       STA $3E
F292   85 51       STA $51        Trackzähler für Formatierung
F294   A9 08       LDA #$08       8
F296   85 39       STA $39        Konstante für Blockheader
F298   A9 07       LDA #$07       7
F29A   85 47       STA $47        Konstante für Datenblock
F29C   A9 05       LDA #$05
F29E   85 62       STA $62
F2A0   A9 FA       LDA #$FA       Zeiger $62/$63 auf $FA05
F2A2   85 63       STA $63
F2A4   A9 C8       LDA #$C8       200
F2A6   85 64       STA $64
F2A8   A9 04       LDA #$04
F2AA   85 5E       STA $5E
F2AC   A9 04       LDA #$04
F2AE   85 5F       STA $5F

****************************** IRQ-Routine für Disk-Controller
F2B0   BA          TSX
F2B1   86 49       STX $49        Stackpointer merken
F2B3   AD 04 1C    LDA $1C04
F2B6   AD 0C 1C    LDA $1C0C      Interruptflag vom Timer löschen
F2B9   09 0E       ORA #$0E
F2BB   8D 0C 1C    STA $1C0C
F2BE   A0 05       LDY #$05
F2C0   B9 00 00    LDA $0000,Y    Auftrag für Puffer Y ?
```

```
F2C3   10 2E      BPL $F2F3     nein
F2C5   C9 D0      CMP #$D0      Kode für Programm im Puffer ausführen ?
F2C7   D0 04      BNE $F2CD     nein
F2C9   98         TYA
F2CA   4C 70 F3   JMP $F370     Programm im Puffer ausführen

F2CD   29 01      AND #$01      Drivenummer isolieren
F2CF   F0 07      BEQ $F2D8     Drive null ?
F2D1   84 3F      STY $3F
F2D3   A9 0F      LDA #$0F      sonst
F2D5   4C 69 F9   JMP $F969     74, 'drive not ready'

F2D8   AA         TAX
F2D9   85 3D      STA $3D
F2DB   C5 3E      CMP $3E       läuft Motor ?
F2DD   F0 0A      BEQ $F2E9     ja
F2DF   20 7E F9   JSR $F97E     Laufwerksmotor einschalten
F2E2   A5 3D      LDA $3D
F2E4   85 3E      STA $3E       Flag setzen
F2E6   4C 9C F9   JMP $F99C     zur Jobschleife

F2E9   A5 20      LDA $20
F2EB   30 03      BMI $F2F0     Kopftransport schon programmiert ?
F2ED   0A         ASL  A
F2EE   10 09      BPL $F2F9
F2F0   4C 9C F9   JMP $F99C     zur Jobschleife

F2F3   88         DEY
F2F4   10 CA      BPL $F2C0     nächsten Puffer prüfen
F2F6   4C 9C F9   JMP $F99C     zur Jobschleife

F2F9   A9 20      LDA #$20
F2FB   85 20      STA $20       Kopftransport programmieren
F2FD   A0 05      LDY #$05
F2FF   84 3F      STY $3F       Pufferzähler initialisieren
F301   20 93 F3   JSR $F393     Zeiger in Puffer setzen
F304   30 1A      BMI $F320     liegt Auftrag für Puffer vor ?
F306   C6 3F      DEC $3F       Zähler erniedrigen
F308   10 F7      BPL $F301     nächsten Puffer prüfen
F30A   A4 41      LDY $41       Puffernummer
F30C   20 95 F3   JSR $F395     Zeiger in Puffer setzen
F30F   A5 42      LDA $42       Trackdifferenz zu letztem Job
F311   85 4A      STA $4A       als Zähler für Kopftransport
F313   06 4A      ASL $4A
F315   A9 60      LDA #$60      Flag für Kopftransport setzen
F317   85 20      STA $20
F319   B1 32      LDA ($32),Y   Tracknummer aus Puffer holen
F31B   85 22      STA $22
F31D   4C 9C F9   JMP $F99C     zur Jobschleife

F320   29 01      AND #$01      Drivenummer isolieren
F322   C5 3D      CMP $3D       gleich Drivenummer des letzten Jobs ?
F324   D0 E0      BNE $F306     nein
F326   A5 22      LDA $22       letzte Tracknummer
F328   F0 12      BEQ $F33C     gleich null ?
F32A   38         SEC
```

```
F32B   F1 32      SBC ($32),Y   gleich der Tracknummer dieses Jobs ?
F32D   F0 0D      BEQ $F33C     ja
F32F   49 FF      EOR #$FF
F331   85 42      STA $42
F333   E6 42      INC $42
F335   A5 3F      LDA $3F       Drivenummer
F337   85 41      STA $41
F339   4C 06 F3   JMP $F306     weiter prüfen

F33C   A2 04      LDX #$04
F33E   B1 32      LDA ($32),Y   Tracknummer des Jobs
F340   85 40      STA $40       merken
F342   DD D6 FE   CMP $FED6,X   mit maximaler Tracknummer vergleichen
F345   CA         DEX
F346   B0 FA      BCS $F342     größer ?
F348   BD D1 FE   LDA $FED1,X   Sektorzahl pro Track holen
F34B   85 43      STA $43       und merken
F34D   8A         TXA
F34E   0A         ASL  A
F34F   0A         ASL  A
F350   0A         ASL  A        Nummer des Spurbereichs mal 32
F351   0A         ASL  A
F352   0A         ASL  A
F353   85 44      STA $44       gibt 0, 32, 64, 96
F355   AD 00 1C   LDA $1C00
F358   29 9F      AND #$9F
F35A   05 44      ORA $44       Steuerbyte für Motor generieren
F35C   8D 00 1C   STA $1C00
F35F   A6 3D      LDX $3D
F361   A5 45      LDA $45       Befehlskode
F363   C9 40      CMP #$40      Kopf positionieren ?
F365   F0 15      BEQ $F37C     ja
F367   C9 60      CMP #$60      Befehlskode für Programm im Puffer ausführen ?
F369   F0 03      BEQ $F36E     ja
F36B   4C B1 F3   JMP $F3B1     Blockheader lesen

******************************  Programm im Puffer ausführen
F36E   A5 3F      LDA $3F       Puffernummer
F370   18         CLC
F371   69 03      ADC #$03      plus 3
F373   85 31      STA $31
F375   A9 00      LDA #$00      gleich Adresse des Puffers
F377   85 30      STA $30
F379   6C 30 00   JMP ($0030)   Programm im Puffer ausführen

******************************  Kopf positionieren
F37C   A9 60      LDA #$60
F37E   85 20      STA $20       Flag für Kopftransport setzen
F380   AD 00 1C   LDA $1C00
F383   29 FC      AND #$FC      Steppermotoren einschalten
F385   8D 00 1C   STA $1C00
F388   A9 A4      LDA #$A4      164
F38A   85 4A      STA $4A       Schrittzähler für Kopftransport
F38C   A9 01      LDA #$01
F38E   85 22      STA $22       Tracknummer
F390   4C 69 F9   JMP $F969     ok
```

```
******************************  Zeiger in Puffer initialisieren
F393   A4 3F      LDY $3F       Puffernummer
F395   B9 00 00   LDA $0000,Y   Befehlskode
F398   48         PHA           merken
F399   10 10      BPL $F3AB
F39B   29 78      AND #$78      Bit 0,1,2 und 7 löschen
F39D   85 45      STA $45
F39F   98         TYA           Puffernummer
F3A0   0A         ASL  A        mal 2
F3A1   69 06      ADC #$06      plus 6
F3A3   85 32      STA $32       gleich Zeiger auf aktuellen Puffer
F3A5   98         TYA           Puffernummer
F3A6   18         CLC
F3A7   69 03      ADC #$03      plus 3
F3A9   85 31      STA $31       gleich Pufferadresse hi
F3AB   A0 00      LDY #$00
F3AD   84 30      STY $30       Pufferadresse lo
F3AF   68         PLA           Befehlskode zurückholen
F3B0   60         RTS

******************************  Blockheader lesen, ID überprüfen
F3B1   A2 5A      LDX #$5A      90
F3B3   86 4B      STX $4B       Zähler
F3B5   A2 00      LDX #$00
F3B7   A9 52      LDA #$52      82
F3B9   85 24      STA $24
F3BB   20 56 F5   JSR $F556     SYNC abwarten
F3BE   50 FE      BVC $F3BE     Byte Ready ?
F3C0   B8         CLV
F3C1   AD 01 1C   LDA $1C01     Daten vom Lesekopf
F3C4   C5 24      CMP $24
F3C6   D0 3F      BNE $F407     20, 'read error'
F3C8   50 FE      BVC $F3C8     Byte Ready ?
F3CA   B8         CLV
F3CB   AD 01 1C   LDA $1C01     Datenbyte von Diskette  (Blockheader)
F3CE   95 25      STA $25,X     7 Bytes speichern
F3D0   E8         INX
F3D1   E0 07      CPX #$07
F3D3   D0 F3      BNE $F3C8     weiter einlesen
F3D5   20 97 F4   JSR $F497
F3D8   A0 04      LDY #$04      4 Byte plus Parity
F3DA   A9 00      LDA #$00
F3DC   59 16 00   EOR $0016,Y   Prüfsumme über Header bilden
F3DF   88         DEY
F3E0   10 FA      BPL $F3DC
F3E2   C9 00      CMP #$00      Parity in Ordnung ?
F3E4   D0 38      BNE $F41E     27,'write error'
F3E6   A6 3E      LDX $3E       Drivenummer
F3E8   A5 18      LDA $18       Tracknummer des Headers
F3EA   95 22      STA $22,X     als aktuelle Tracknummer übernehmen
F3EC   A5 45      LDA $45
F3EE   C9 30      CMP #$30      Kode für 'Header übernehmen'
F3F0   F0 1E      BEQ $F410     Header übernehmen
F3F2   A5 3E      LDA $3E
F3F4   0A         ASL  A
F3F5   A8         TAY
```

```
F3F6   B9 12 00   LDA $0012,Y
F3F9   C5 16      CMP $16       ID1 vergleichen
F3FB   D0 1E      BNE $F41B
F3FD   B9 13 00   LDA $0013,Y
F400   C5 17      CMP $17       ID2 vergleichen
F402   D0 17      BNE $F41B     ungleich, dann 29, 'disk id mismatch'
F404   4C 23 F4   JMP $F423

F407   C6 4B      DEC $4B       Zähler für Versuche erniedrigen
F409   D0 B0      BNE $F3BB     und nochmal probieren
F40B   A9 02      LDA #$02      ansonsten
F40D   20 69 F9   JSR $F969     20, 'read error'

******************************  Blockheader übernehmen
F410   A5 16      LDA $16       ID1
F412   85 12      STA $12
F414   A5 17      LDA $17       und ID2
F416   85 13      STA $13       übernehmen
F418   A9 01      LDA #$01      ok
F41A   2C         .BYTE $2C
F41B   A9 0B      LDA #$0B      29, 'disk id mismatch'
F41D   2C         .BYTE $2C
F41E   A9 09      LDA #$09      27, 'write error'
F420   4C 69 F9   JMP $F969     Abschluß

******************************
F423   A9 7F      LDA #$7F
F425   85 4C      STA $4C
F427   A5 19      LDA $19
F429   18         CLC
F42A   69 02      ADC #$02
F42C   C5 43      CMP $43
F42E   90 02      BCC $F432
F430   E5 43      SBC $43
F432   85 4D      STA $4D
F434   A2 05      LDX #$05
F436   86 3F      STX $3F
F438   A2 FF      LDX #$FF
F43A   20 93 F3   JSR $F393     Pufferzeiger für Disk-Controller setzen
F43D   10 44      BPL $F483
F43F   85 44      STA $44
F441   29 01      AND #$01
F443   C5 3E      CMP $3E
F445   D0 3C      BNE $F483
F447   A0 00      LDY #$00
F449   B1 32      LDA ($32),Y
F44B   C5 40      CMP $40
F44D   D0 34      BNE $F483
F44F   A5 45      LDA $45       Befehlskode
F451   C9 60      CMP #$60
F453   F0 0C      BEQ $F461
F455   A0 01      LDY #$01
F457   38         SEC
F458   B1 32      LDA ($32),Y
F45A   E5 4D      SBC $4D
F45C   10 03      BPL $F461
```

```
F45E   18         CLC
F45F   65 43      ADC $43
F461   C5 4C      CMP $4C
F463   B0 1E      BCS $F483
F465   48         PHA
F466   A5 45      LDA $45
F468   F0 14      BEQ $F47E
F46A   68         PLA
F46B   C9 09      CMP #$09
F46D   90 14      BCC $F483
F46F   C9 0C      CMP #$0C
F471   B0 10      BCS $F483
F473   85 4C      STA $4C
F475   A5 3F      LDA $3F
F477   AA         TAX
F478   69 03      ADC #$03
F47A   85 31      STA $31
F47C   D0 05      BNE $F483
F47E   68         PLA
F47F   C9 06      CMP #$06
F481   90 F0      BCC $F473
F483   C6 3F      DEC $3F
F485   10 B3      BPL $F43A
F487   8A         TXA
F488   10 03      BPL $F48D
F48A   4C 9C F9   JMP $F99C     zur Jobschleife

F48D   86 3F      STX $3F
F48F   20 93 F3   JSR $F393     Puffernummer holen
F492   A5 45      LDA $45       Befehlskode
F494   4C CA F4   JMP $F4CA     weiter prüfen

F497   A5 30      LDA $30
F499   48         PHA           Zeiger $30/$31 retten
F49A   A5 31      LDA $31
F49C   48         PHA
F49D   A9 24      LDA #$24
F49F   85 30      STA $30
F4A1   A9 00      LDA #$00      Zeiger $30/$31 auf $24
F4A3   85 31      STA $31
F4A5   A9 00      LDA #$00
F4A7   85 34      STA $34
F4A9   20 E6 F7   JSR $F7E6
F4AC   A5 55      LDA $55
F4AE   85 18      STA $18
F4B0   A5 54      LDA $54
F4B2   85 19      STA $19
F4B4   A5 53      LDA $53
F4B6   85 1A      STA $1A
F4B8   20 E6 F7   JSR $F7E6
F4BB   A5 52      LDA $52
F4BD   85 17      STA $17
F4BF   A5 53      LDA $53
F4C1   85 16      STA $16
F4C3   68         PLA
F4C4   85 31      STA $31
```

```
F4C6   68          PLA            Zeiger $30/$31 zurückholen
F4C7   85 30       STA $30
F4C9   60          RTS

*******************************
F4CA   C9 00       CMP #$00       Befehlskode für 'Lesen' ?
F4CC   F0 03       BEQ $F4D1      ja
F4CE   4C 6E F5    JMP $F56E      Befehlskode weiter prüfen

F4D1   20 0A F5    JSR $F50A      Datenblockanfang suchen
F4D4   50 FE       BVC $F4D4      Byte Ready ?
F4D6   B8          CLV
F4D7   AD 01 1C    LDA $1C01      Datenbyte holen
F4DA   91 30       STA ($30),Y    und in Puffer schreiben
F4DC   C8          INY            256 mal
F4DD   D0 F5       BNE $F4D4
F4DF   A0 BA       LDY #$BA
F4E1   50 FE       BVC $F4E1      Byte Ready ?
F4E3   B8          CLV
F4E4   AD 01 1C    LDA $1C01      Bytes lesen
F4E7   99 00 01    STA $0100,Y    nach $1BA bis $1FF
F4EA   C8          INY
F4EB   D0 F4       BNE $F4E1
F4ED   20 E0 F8    JSR $F8E0
F4F0   A5 38       LDA $38
F4F2   C5 47       CMP $47        gleich 7, Beginn Datenblock ?
F4F4   F0 05       BEQ $F4FB      ja
F4F6   A9 04       LDA #$04       22, 'read error'
F4F8   4C 69 F9    JMP $F969      Fehlerabschluß

F4FB   20 E9 F5    JSR $F5E9      Parity des Datenblock berechnen
F4FE   C5 3A       CMP $3A        Übereinstimmung ?
F500   F0 03       BEQ $F505      ja
F502   A9 05       LDA #$05       23, 'read error'
F504   2C          .BYTE $2C
F505   A9 01       LDA #$01       ok
F507   4C 69 F9    JMP $F969      Fehlermeldung bereitstellen

*******************************   Datenblockanfang suchen
F50A   20 10 F5    JSR $F510      Blockheader lesen
F50D   4C 56 F5    JMP $F556      SYNC abwarten

*******************************   Blockheader lesen
F510   A5 3D       LDA $3D        Drivenummer
F512   0A          ASL  A
F513   AA          TAX
F514   B5 12       LDA $12,X      ID1
F516   85 16       STA $16        merken
F518   B5 13       LDA $13,X      ID2
F51A   85 17       STA $17        merken
F51C   A0 00       LDY #$00
F51E   B1 32       LDA ($32),Y    Track
F520   85 18       STA $18
F522   C8          INY
F523   B1 32       LDA ($32),Y    und Sektornummer aus Puffer holen
F525   85 19       STA $19
```

```
F527   A9 00      LDA #$00
F529   45 16      EOR $16
F52B   45 17      EOR $17        Parity für Blockheader berechnen
F52D   45 18      EOR $18
F52F   45 19      EOR $19
F531   85 1A      STA $1A        und merken
F533   20 34 F9   JSR $F934
F536   A2 5A      LDX #$5A       90 Versuche
F538   20 56 F5   JSR $F556       SYNC abwarten
F53B   A0 00      LDY #$00
F53D   50 FE      BVC $F53D      Byte Ready ?
F53F   B8         CLV
F540   AD 01 1C   LDA $1C01      Daten vom Blockheader lesen
F543   D9 24 00   CMP $0024,Y    mit gespeicherten Daten vergleichen
F546   D0 06      BNE $F54E      ungleich, dann nochmal versuchen
F548   C8         INY
F549   C0 08      CPY #$08       schon 8 Bytes gelesen ?
F54B   D0 F0      BNE $F53D      nein
F54D   60         RTS

F54E   CA         DEX            Zähler erniedrigen
F54F   D0 E7      BNE $F538      noch nicht null ?
F551   A9 02      LDA #$02
F553   4C 69 F9   JMP $F969      20, 'read error'

******************************   SYNC abwarten
F556   A9 D0      LDA #$D0       208
F558   8D 05 18   STA $1805      Timer starten
F55B   A9 03      LDA #$03       Fehlerkode
F55D   2C 05 18   BIT $1805
F560   10 F1      BPL $F553      Timer abgelaufen, dann 21 , 'read error'
F562   2C 00 1C   BIT $1C00      SYNC-Signal
F565   30 F6      BMI $F55D      nocht nicht gefunden ?
F567   AD 01 1C   LDA $1C01      Byte lesen
F56A   B8         CLV
F56B   A0 00      LDY #$00
F56D   60         RTS

******************************
F56E   C9 10      CMP #$10       Befehlskode für 'Schreiben'
F570   F0 03      BEQ $F575      ja
F572   4C 91 F6   JMP $F691      Befehlskode weiter prüfen

******************************   Datenblock auf Diskette schreiben
F575   20 E9 F5   JSR $F5E9      Parity für Puffer berechnen
F578   85 3A      STA $3A        und speichern
F57A   AD 00 1C   LDA $1C00      Port B lesen
F57D   29 10      AND #$10       Bit für 'Write Protect' isolieren
F57F   D0 05      BNE $F586      nicht gesetzt, ok
F581   A9 08      LDA #$08
F583   4C 69 F9   JMP $F969      26, 'write protect on'

F586   20 8F F7   JSR $F78F
F589   20 10 F5   JSR $F510      Blockheader suchen
F58C   A2 09      LDX #$09
F58E   50 FE      BVC $F58E      Byte Ready ?
```

```
F590   B8          CLV
F591   CA          DEX             9 Bytes nach Blockheader überlesen
F592   D0 FA       BNE $F58E
F594   A9 FF       LDA #$FF
F596   8D 03 1C    STA $1C03       Port A (Schreib/Lesekopf) auf Ausgang
F599   AD 0C 1C    LDA $1C0C
F59C   29 1F       AND #$1F
F59E   09 C0       ORA #$C0        PCR auf Ausgabe umschalten
F5A0   8D 0C 1C    STA $1C0C
F5A3   A9 FF       LDA #$FF
F5A5   A2 05       LDX #$05
F5A7   8D 01 1C    STA $1C01       5 mal $FF auf Diskette schreiben
F5AA   B8          CLV
F5AB   50 FE       BVC $F5AB       als SYNC-Zeichen
F5AD   B8          CLV
F5AE   CA          DEX
F5AF   D0 FA       BNE $F5AB
F5B1   A0 BB       LDY #$BB
F5B3   B9 00 01    LDA $0100,Y     Bytes $1BB bis $1FF auf Diskette
F5B6   50 FE       BVC $F5B6
F5B8   B8          CLV
F5B9   8D 01 1C    STA $1C01
F5BC   C8          INY
F5BD   D0 F4       BNE $F5B3
F5BF   B1 30       LDA ($30),Y     Datenpuffer (256 Bytes) auf Diskette schreiben
F5C1   50 FE       BVC $F5C1
F5C3   B8          CLV
F5C4   8D 01 1C    STA $1C01
F5C7   C8          INY
F5C8   D0 F5       BNE $F5BF
F5CA   50 FE       BVC $F5CA       Byte Ready ?
F5CC   AD 0C 1C    LDA $1C0C
F5CF   09 E0       ORA #$E0        PCR wieder auf Eingabe
F5D1   8D 0C 1C    STA $1C0C
F5D4   A9 00       LDA #$00
F5D6   8D 03 1C    STA $1C03       Port A (Schreib/Lesekopf) auf Eingang
F5D9   20 F2 F5    JSR $F5F2
F5DC   A4 3F       LDY $3F
F5DE   B9 00 00    LDA $0000,Y
F5E1   49 30       EOR #$30        Befehlskode 'Schreiben' in 'Verify' umwandeln
F5E3   99 00 00    STA $0000,Y
F5E6   4C B1 F3    JMP $F3B1

****************************       Parity für Datenpuffer berechnen
F5E9   A9 00       LDA #$00
F5EB   A8          TAY
F5EC   51 30       EOR ($30),Y
F5EE   C8          INY
F5EF   D0 FB       BNE $F5EC
F5F1   60          RTS

F5F2   A9 00       LDA #$00
F5F4   85 2E       STA $2E
F5F6   85 30       STA $30
F5F8   85 4F       STA $4F
F5FA   A5 31       LDA $31
```

```
F5FC   85 4E      STA $4E
F5FE   A9 01      LDA #$01
F600   85 31      STA $31
F602   85 2F      STA $2F
F604   A9 BB      LDA #$BB
F606   85 34      STA $34
F608   85 36      STA $36
F60A   20 E6 F7   JSR $F7E6
F60D   A5 52      LDA $52
F60F   85 38      STA $38
F611   A4 36      LDY $36
F613   A5 53      LDA $53
F615   91 2E      STA ($2E),Y
F617   C8         INY
F618   A5 54      LDA $54
F61A   91 2E      STA ($2E),Y
F61C   C8         INY
F61D   A5 55      LDA $55
F61F   91 2E      STA ($2E),Y
F621   C8         INY
F622   84 36      STY $36
F624   20 E6 F7   JSR $F7E6
F627   A4 36      LDY $36
F629   A5 52      LDA $52
F62B   91 2E      STA ($2E),Y
F62D   C8         INY
F62E   A5 53      LDA $53
F630   91 2E      STA ($2E),Y
F632   C8         INY
F633   F0 0E      BEQ $F643
F635   A5 54      LDA $54
F637   91 2E      STA ($2E),Y
F639   C8         INY
F63A   A5 55      LDA $55
F63C   91 2E      STA ($2E),Y
F63E   C8         INY
F63F   84 36      STY $36
F641   D0 E1      BNE $F624
F643   A5 54      LDA $54
F645   91 30      STA ($30),Y
F647   C8         INY
F648   A5 55      LDA $55
F64A   91 30      STA ($30),Y
F64C   C8         INY
F64D   84 36      STY $36
F64F   20 E6 F7   JSR $F7E6
F652   A4 36      LDY $36
F654   A5 52      LDA $52
F656   91 30      STA ($30),Y
F658   C8         INY
F659   A5 53      LDA $53
F65B   91 30      STA ($30),Y
F65D   C8         INY
F65E   A5 54      LDA $54
F660   91 30      STA ($30),Y
F662   C8         INY
```

```
F663   A5 55      LDA $55
F665   91 30      STA ($30),Y
F667   C8         INY
F668   84 36      STY $36
F66A   C0 BB      CPY #$BB
F66C   90 E1      BCC $F64F
F66E   A9 45      LDA #$45
F670   85 2E      STA $2E
F672   A5 31      LDA $31
F674   85 2F      STA $2F
F676   A0 BA      LDY #$BA
F678   B1 30      LDA ($30),Y
F67A   91 2E      STA ($2E),Y
F67C   88         DEY
F67D   D0 F9      BNE $F678
F67F   B1 30      LDA ($30),Y
F681   91 2E      STA ($2E),Y
F683   A2 BB      LDX #$BB
F685   BD 00 01   LDA $0100,X
F688   91 30      STA ($30),Y
F68A   C8         INY
F68B   E8         INX
F68C   D0 F7      BNE $F685
F68E   86 50      STX $50
F690   60         RTS
*******************************
F691   C9 20      CMP #$20      Befehlskode für 'Verify' ?
F693   F0 03      BEQ $F698     ja
F695   4C CA F6   JMP $F6CA     Befehlskode weiter prüfen

F698   20 E9 F5   JSR $F5E9     Parity für Datenpuffer berechnen
F69B   85 3A      STA $3A       und merken
F69D   20 8F F7   JSR $F78F
F6A0   20 0A F5   JSR $F50A     Datenblockanfang suchen
F6A3   A0 BB      LDY #$BB
F6A5   B9 00 01   LDA $0100,Y   Daten aus Puffer
F6A8   50 FE      BVC $F6A8     Byte Ready ?
F6AA   B8         CLV
F6AB   4D 01 1C   EOR $1C01     mit Daten von Diskette vergleichen
F6AE   D0 15      BNE $F6C5     ungleich, dann Fehler
F6B0   C8         INY
F6B1   D0 F2      BNE $F6A5
F6B3   B1 30      LDA ($30),Y   Daten aus Puffer
F6B5   50 FE      BVC $F6B5
F6B7   B8         CLV
F6B8   4D 01 1C   EOR $1C01     mit Daten von Diskette vergleichen
F6BB   D0 08      BNE $F6C5     ungleich, dann Fehler
F6BD   C8         INY
F6BE   C0 FD      CPY #$FD
F6C0   D0 F1      BNE $F6B3
F6C2   4C 18 F4   JMP $F418     fehlerfreier Abschluß

F6C5   A9 07      LDA #$07
F6C7   4C 69 F9   JMP $F969     25, 'write error'

*******************************
```

```
F6CA   20 10 F5   JSR $F510      Blockheader lesen
F6CD   4C 18 F4   JMP $F418      fertig

*****************************
F6D0   A9 00      LDA #$00
F6D2   85 57      STA $57
F6D4   85 5A      STA $5A
F6D6   A4 34      LDY $34
F6D8   A5 52      LDA $52
F6DA   29 F0      AND #$F0       Hi-Nibble isolieren
F6DC   4A         LSR  A
F6DD   4A         LSR  A         und in unteres Nibble schieben
F6DE   4A         LSR  A
F6DF   4A         LSR  A
F6E0   AA         TAX            als Index in Tabelle
F6E1   BD 7F F7   LDA $F77F,X
F6E4   0A         ASL  A
F6E5   0A         ASL  A         mal 8
F6E6   0A         ASL  A
F6E7   85 56      STA $56
F6E9   A5 52      LDA $52
F6EB   29 0F      AND #$0F       unteres Nibble isolieren
F6ED   AA         TAX            als Index in Tabelle
F6EE   BD 7F F7   LDA $F77F,X
F6F1   6A         ROR  A
F6F2   66 57      ROR $57
F6F4   6A         ROR  A
F6F5   66 57      ROR $57
F6F7   29 07      AND #$07
F6F9   05 56      ORA $56
F6FB   91 30      STA ($30),Y    in Puffer
F6FD   C8         INY            Pufferzeiger erhöhen
F6FE   A5 53      LDA $53
F700   29 F0      AND #$F0       oberes Nibble isolieren
F702   4A         LSR  A
F703   4A         LSR  A
F704   4A         LSR  A         in unteres Nibble schieben
F705   4A         LSR  A
F706   AA         TAX            als Index in Tabelle
F707   BD 7F F7   LDA $F77F,X
F70A   0A         ASL  A
F70B   05 57      ORA $57
F70D   85 57      STA $57
F70F   A5 53      LDA $53
F711   29 0F      AND #$0F       unteres Nibble
F713   AA         TAX            als Index
F714   BD 7F F7   LDA $F77F,X
F717   2A         ROL  A
F718   2A         ROL  A
F719   2A         ROL  A
F71A   2A         ROL  A
F71B   85 58      STA $58
F71D   2A         ROL  A
F71E   29 01      AND #$01
F720   05 57      ORA $57
F722   91 30      STA ($30),Y    in Puffer
```

```
F724   C8          INY             Pufferzeiger erhöhen
F725   A5 54       LDA $54
F727   29 F0       AND #$F0        Hi-Nibble isolieren
F729   4A          LSR  A
F72A   4A          LSR  A
F72B   4A          LSR  A
F72C   4A          LSR  A
F72D   AA          TAX
F72E   BD 7F F7    LDA $F77F,X
F731   18          CLC
F732   6A          ROR  A
F733   05 58       ORA $58
F735   91 30       STA ($30),Y     in Puffer
F737   C8          INY             Pufferzeiger erhöhen
F738   6A          ROR  A
F739   29 80       AND #$80
F73B   85 59       STA $59
F73D   A5 54       LDA $54
F73F   29 0F       AND #$0F        unteres Nibble
F741   AA          TAX             als Index
F742   BD 7F F7    LDA $F77F,X
F745   0A          ASL  A
F746   0A          ASL  A
F747   29 7C       AND #$7C
F749   05 59       ORA $59
F74B   85 59       STA $59
F74D   A5 55       LDA $55
F74F   29 F0       AND #$F0        Hi-Nibble isolieren
F751   4A          LSR  A
F752   4A          LSR  A          in unteres Nibble schieben
F753   4A          LSR  A
F754   4A          LSR  A
F755   AA          TAX             als Index in Tabelle
F756   BD 7F F7    LDA $F77F,X
F759   6A          ROR  A
F75A   66 5A       ROR $5A
F75C   6A          ROR  A
F75D   66 5A       ROR $5A
F75F   6A          ROR  A
F760   66 5A       ROR $5A
F762   29 03       AND #$03
F764   05 59       ORA $59
F766   91 30       STA ($30),Y     in Puffer
F768   C8          INY             Pufferzeiger erhöhen
F769   D0 04       BNE $F76F
F76B   A5 2F       LDA $2F
F76D   85 31       STA $31
F76F   A5 55       LDA $55
F771   29 0F       AND #$0F        unteres Nibble
F773   AA          TAX             als Index
F774   BD 7F F7    LDA $F77F,X
F777   05 5A       ORA $5A
F779   91 30       STA ($30),Y     in Puffer
F77B   C8          INY             Pufferzeiger erhöhen
F77C   84 34       STY $34         und merken
F77E   60          RTS
```

```
******************************
F77F 0A 0B 12 13 0E 0F 16 17
F787 09 19 1A 1B 0D 1D 1E 15

******************************
F78F   A9 00      LDA #$00
F791   85 30      STA $30
F793   85 2E      STA $2E
F795   85 36      STA $36
F797   A9 BB      LDA #$BB
F799   85 34      STA $34
F79B   85 50      STA $50
F79D   A5 31      LDA $31
F79F   85 2F      STA $2F
F7A1   A9 01      LDA #$01
F7A3   85 31      STA $31
F7A5   A5 47      LDA $47
F7A7   85 52      STA $52
F7A9   A4 36      LDY $36
F7AB   B1 2E      LDA ($2E),Y
F7AD   85 53      STA $53
F7AF   C8         INY
F7B0   B1 2E      LDA ($2E),Y
F7B2   85 54      STA $54
F7B4   C8         INY
F7B5   B1 2E      LDA ($2E),Y
F7B7   85 55      STA $55
F7B9   C8         INY
F7BA   84 36      STY $36
F7BC   20 D0 F6   JSR $F6D0
F7BF   A4 36      LDY $36
F7C1   B1 2E      LDA ($2E),Y
F7C3   85 52      STA $52
F7C5   C8         INY
F7C6   F0 11      BEQ $F7D9
F7C8   B1 2E      LDA ($2E),Y
F7CA   85 53      STA $53
F7CC   C8         INY
F7CD   B1 2E      LDA ($2E),Y
F7CF   85 54      STA $54
F7D1   C8         INY
F7D2   B1 2E      LDA ($2E),Y
F7D4   85 55      STA $55
F7D6   C8         INY
F7D7   D0 E1      BNE $F7BA
F7D9   A5 3A      LDA $3A
F7DB   85 53      STA $53
F7DD   A9 00      LDA #$00
F7DF   85 54      STA $54
F7E1   85 55      STA $55
F7E3   4C D0 F6   JMP $F6D0

F7E6   A4 34      LDY $34
F7E8   B1 30      LDA ($30),Y
F7EA   29 F8      AND #$F8
F7EC   4A         LSR  A
```

```
F7ED   4A        LSR  A
F7EE   4A        LSR  A
F7EF   85 56     STA $56
F7F1   B1 30     LDA ($30),Y
F7F3   29 07     AND #$07
F7F5   0A        ASL  A
F7F6   0A        ASL  A
F7F7   85 57     STA $57
F7F9   C8        INY
F7FA   D0 06     BNE $F802
F7FC   A5 4E     LDA $4E
F7FE   85 31     STA $31
F800   A4 4F     LDY $4F
F802   B1 30     LDA ($30),Y
F804   29 C0     AND #$C0
F806   2A        ROL  A
F807   2A        ROL  A
F808   2A        ROL  A
F809   05 57     ORA $57
F80B   85 57     STA $57
F80D   B1 30     LDA ($30),Y
F80F   29 3E     AND #$3E
F811   4A        LSR  A
F812   85 58     STA $58
F814   B1 30     LDA ($30),Y
F816   29 01     AND #$01
F818   0A        ASL  A
F819   0A        ASL  A
F81A   0A        ASL  A
F81B   0A        ASL  A
F81C   85 59     STA $59
F81E   C8        INY
F81F   B1 30     LDA ($30),Y
F821   29 F0     AND #$F0
F823   4A        LSR  A
F824   4A        LSR  A
F825   4A        LSR  A
F826   4A        LSR  A
F827   05 59     ORA $59
F829   85 59     STA $59
F82B   B1 30     LDA ($30),Y
F82D   29 0F     AND #$0F
F82F   0A        ASL  A
F830   85 5A     STA $5A
F832   C8        INY
F833   B1 30     LDA ($30),Y
F835   29 80     AND #$80
F837   18        CLC
F838   2A        ROL  A
F839   2A        ROL  A
F83A   29 01     AND #$01
F83C   05 5A     ORA $5A
F83E   85 5A     STA $5A
F840   B1 30     LDA ($30),Y
F842   29 7C     AND #$7C
F844   4A        LSR  A
```

```
F845   4A          LSR  A
F846   85 5B       STA $5B
F848   B1 30       LDA ($30),Y
F84A   29 03       AND #$03
F84C   0A          ASL  A
F84D   0A          ASL  A
F84E   0A          ASL  A
F84F   85 5C       STA $5C
F851   C8          INY
F852   D0 06       BNE $F85A
F854   A5 4E       LDA $4E
F856   85 31       STA $31
F858   A4 4F       LDY $4F
F85A   B1 30       LDA ($30),Y
F85C   29 E0       AND #$E0
F85E   2A          ROL  A
F85F   2A          ROL  A
F860   2A          ROL  A
F861   2A          ROL  A
F862   05 5C       ORA $5C
F864   85 5C       STA $5C
F866   B1 30       LDA ($30),Y
F868   29 1F       AND #$1F
F86A   85 5D       STA $5D
F86C   C8          INY
F86D   84 34       STY $34
F86F   A6 56       LDX $56
F871   BD A0 F8    LDA $F8A0,X
F874   A6 57       LDX $57
F876   1D C0 F8    ORA $F8C0,X
F879   85 52       STA $52
F87B   A6 58       LDX $58
F87D   BD A0 F8    LDA $F8A0,X
F880   A6 59       LDX $59
F882   1D C0 F8    ORA $F8C0,X
F885   85 53       STA $53
F887   A6 5A       LDX $5A
F889   BD A0 F8    LDA $F8A0,X
F88C   A6 5B       LDX $5B
F88E   1D C0 F8    ORA $F8C0,X
F891   85 54       STA $54
F893   A6 5C       LDX $5C
F895   BD A0 F8    LDA $F8A0,X
F898   A6 5D       LDX $5D
F89A   1D C0 F8    ORA $F8C0,X
F89D   85 55       STA $55
F89F   60          RTS

******************************
F8A0 FF FF FF FF FF FF FF FF
F8A8 FF 80 00 10 FF C0 40 50
F8B0 FF FF 20 30 FF F0 60 70
F8B8 FF 90 A0 B0 FF D0 E0 FF

F8C0 FF FF FF FF FF FF FF FF
F8C8 FF 08 00 01 FF 0C 04 05
```

```
F8D0 FF FF 02 03 FF 0F 06 07
F8D8 FF 09 0A 0B FF 0D 0E FF

******************************
F8E0   A9 00      LDA #$00
F8E2   85 34      STA $34
F8E4   85 2E      STA $2E
F8E6   85 36      STA $36
F8E8   A9 01      LDA #$01
F8EA   85 4E      STA $4E
F8EC   A9 BA      LDA #$BA
F8EE   85 4F      STA $4F
F8F0   A5 31      LDA $31
F8F2   85 2F      STA $2F
F8F4   20 E6 F7   JSR $F7E6
F8F7   A5 52      LDA $52
F8F9   85 38      STA $38
F8FB   A4 36      LDY $36
F8FD   A5 53      LDA $53
F8FF   91 2E      STA ($2E),Y
F901   C8         INY
F902   A5 54      LDA $54
F904   91 2E      STA ($2E),Y
F906   C8         INY
F907   A5 55      LDA $55
F909   91 2E      STA ($2E),Y
F90B   C8         INY
F90C   84 36      STY $36
F90E   20 E6 F7   JSR $F7E6
F911   A4 36      LDY $36
F913   A5 52      LDA $52
F915   91 2E      STA ($2E),Y
F917   C8         INY
F918   F0 11      BEQ $F92B
F91A   A5 53      LDA $53
F91C   91 2E      STA ($2E),Y
F91E   C8         INY
F91F   A5 54      LDA $54
F921   91 2E      STA ($2E),Y
F923   C8         INY
F924   A5 55      LDA $55
F926   91 2E      STA ($2E),Y
F928   C8         INY
F929   D0 E1      BNE $F90C
F92B   A5 53      LDA $53
F92D   85 3A      STA $3A
F92F   A5 2F      LDA $2F
F931   85 31      STA $31
F933   60         RTS

F934   A5 31      LDA $31
F936   85 2F      STA $2F
F938   A9 00      LDA #$00
F93A   85 31      STA $31
F93C   A9 24      LDA #$24
F93E   85 34      STA $34
```

```
F940   A5 39      LDA $39
F942   85 52      STA $52
F944   A5 1A      LDA $1A
F946   85 53      STA $53
F948   A5 19      LDA $19
F94A   85 54      STA $54
F94C   A5 18      LDA $18
F94E   85 55      STA $55
F950   20 D0 F6   JSR $F6D0
F953   A5 17      LDA $17
F955   85 52      STA $52
F957   A5 16      LDA $16
F959   85 53      STA $53
F95B   A9 00      LDA #$00
F95D   85 54      STA $54
F95F   85 55      STA $55
F961   20 D0 F6   JSR $F6D0
F964   A5 2F      LDA $2F
F966   85 31      STA $31
F968   60         RTS

F969   A4 3F      LDY $3F
F96B   99 00 00   STA $0000,Y
F96E   A5 50      LDA $50
F970   F0 03      BEQ $F975
F972   20 F2 F5   JSR $F5F2
F975   20 8F F9   JSR $F98F
F978   A6 49      LDX $49       Stackpointer zurückholen
F97A   9A         TXS
F97B   4C BE F2   JMP $F2BE

F97E   A9 A0      LDA #$A0
F980   85 20      STA $20
F982   AD 00 1C   LDA $1C00
F985   09 04      ORA #$04      Laufwerkmotor ausschalten
F987   8D 00 1C   STA $1C00
F98A   A9 3C      LDA #$3C
F98C   85 48      STA $48
F98E   60         RTS

F98F   A6 3E      LDX $3E
F991   A5 20      LDA $20
F993   09 10      ORA #$10
F995   85 20      STA $20
F997   A9 FF      LDA #$FF
F999   85 48      STA $48
F99B   60         RTS

F99C   AD 07 1C   LDA $1C07
F99F   8D 05 1C   STA $1C05
F9A2   AD 00 1C   LDA $1C00
F9A5   29 10      AND #$10      'Write Protect' ?
F9A7   C5 1E      CMP $1E
F9A9   85 1E      STA $1E
F9AB   F0 04      BEQ $F9B1
F9AD   A9 01      LDA #$01
```

```
F9AF   85 1C      STA $1C
F9B1   AD FE 02   LDA $02FE
F9B4   F0 15      BEQ $F9CB
F9B6   C9 02      CMP #$02
F9B8   D0 07      BNE $F9C1
F9BA   A9 00      LDA #$00
F9BC   8D FE 02   STA $02FE
F9BF   F0 0A      BEQ $F9CB
F9C1   85 4A      STA $4A
F9C3   A9 02      LDA #$02
F9C5   8D FE 02   STA $02FE
F9C8   4C 2E FA   JMP $FA2E

F9CB   A6 3E      LDX $3E
F9CD   30 07      BMI $F9D6
F9CF   A5 20      LDA $20
F9D1   A8         TAY
F9D2   C9 20      CMP #$20
F9D4   D0 03      BNE $F9D9
F9D6   4C BE FA   JMP $FABE

F9D9   C6 48      DEC $48
F9DB   D0 1D      BNE $F9FA
F9DD   98         TYA
F9DE   10 04      BPL $F9E4
F9E0   29 7F      AND #$7F
F9E2   85 20      STA $20
F9E4   29 10      AND #$10
F9E6   F0 12      BEQ $F9FA
F9E8   AD 00 1C   LDA $1C00
F9EB   29 FB      AND #$FB      Laufwerkmotor ein
F9ED   8D 00 1C   STA $1C00
F9F0   A9 FF      LDA #$FF
F9F2   85 3E      STA $3E
F9F4   A9 00      LDA #$00
F9F6   85 20      STA $20
F9F8   F0 DC      BEQ $F9D6
F9FA   98         TYA
F9FB   29 40      AND #$40
F9FD   D0 03      BNE $FA02
F9FF   4C BE FA   JMP $FABE

FA02   6C 62 00   JMP ($0062)

FA05   A5 4A      LDA $4A
FA07   10 05      BPL $FA0E
FA09   49 FF      EOR #$FF
FA0B   18         CLC
FA0C   69 01      ADC #$01
FA0E   C5 64      CMP $64
FA10   B0 0A      BCS $FA1C
FA12   A9 3B      LDA #$3B
FA14   85 62      STA $62
FA16   A9 FA      LDA #$FA      Zeiger $62/$63 auf $FA3B
FA18   85 63      STA $63
FA1A   D0 12      BNE $FA2E
```

```
FA1C   E5 5E      SBC $5E
FA1E   E5 5E      SBC $5E
FA20   85 61      STA $61
FA22   A5 5E      LDA $5E
FA24   85 60      STA $60
FA26   A9 7B      LDA #$7B
FA28   85 62      STA $62
FA2A   A9 FA      LDA #$FA      Zeiger $62/$63 auf $FA7B
FA2C   85 63      STA $63
FA2E   A5 4A      LDA $4A       Schrittzähler für Kopftransport
FA30   10 31      BPL $FA63
FA32   E6 4A      INC $4A       erhöhen
FA34   AE 00 1C   LDX $1C00
FA37   CA         DEX
FA38   4C 69 FA   JMP $FA69

*****************************
FA3B   A5 4A      LDA $4A       Schrittzähler für Kopftransport
FA3D   D0 EF      BNE $FA2E     noch nicht null ?
FA3F   A9 4E      LDA #$4E
FA41   85 62      STA $62
FA43   A9 FA      LDA #$FA      Zeiger $62/$63 auf $FA4E
FA45   85 63      STA $63
FA47   A9 05      LDA #$05
FA49   85 60      STA $60       Zähler auf 5
FA4B   4C BE FA   JMP $FABE

*****************************
FA4E   C6 60      DEC $60       Zähler erniedrigen
FA50   D0 6C      BNE $FABE     noch nicht null ?
FA52   A5 20      LDA $20
FA54   29 BF      AND #$BF      Bit 6 löschen
FA56   85 20      STA $20
FA58   A9 05      LDA #$05
FA5A   85 62      STA $62
FA5C   A9 FA      LDA #$FA      Zeiger $62/$63 auf $FA05
FA5E   85 63      STA $63
FA60   4C BE FA   JMP $FABE

*****************************
FA63   C6 4A      DEC $4A       Schrittzähler für Kopftransport erniedrigen
FA65   AE 00 1C   LDX $1C00
FA68   E8         INX
FA69   8A         TXA
FA6A   29 03      AND #$03
FA6C   85 4B      STA $4B
FA6E   AD 00 1C   LDA $1C00
FA71   29 FC      AND #$FC
FA73   05 4B      ORA $4B       Steppermotor aus
FA75   8D 00 1C   STA $1C00
FA78   4C BE FA   JMP $FABE

*****************************
FA7B   38         SEC
FA7C   AD 07 1C   LDA $1C07
FA7F   E5 5F      SBC $5F
```

```
FA81   8D 05 1C   STA $1C05
FA84   C6 60      DEC $60       Zähler erniedrigen
FA86   D0 0C      BNE $FA94     noch nicht null ?
FA88   A5 5E      LDA $5E
FA8A   85 60      STA $60       Zähler neu setzen
FA8C   A9 97      LDA #$97
FA8E   85 62      STA $62
FA90   A9 FA      LDA #$FA      Zeiger $62/$63 auf $FA97
FA92   85 63      STA $63
FA94   4C 2E FA   JMP $FA2E

******************************
FA97   C6 61      DEC $61
FA99   D0 F9      BNE $FA94
FA9B   A9 A5      LDA #$A5
FA9D   85 62      STA $62
FA9F   A9 FA      LDA #$FA      Zeiger $62/$63 auf $FAA5
FAA1   85 63      STA $63
FAA3   D0 EF      BNE $FA94

******************************
FAA5   AD 07 1C   LDA $1C07
FAA8   18         CLC
FAA9   65 5F      ADC $5F
FAAB   8D 05 1C   STA $1C05
FAAE   C6 60      DEC $60       Zähler erniedrigen
FAB0   D0 E2      BNE $FA94     noch nicht null ?
FAB2   A9 4E      LDA #$4E
FAB4   85 62      STA $62
FAB6   A9 FA      LDA #$FA      Zeiger $62/$63 auf $FA4E
FAB8   85 63      STA $63
FABA   A9 05      LDA #$05
FABC   85 60      STA $60       Zähler auf 5
FABE   AD 0C 1C   LDA $1C0C
FAC1   29 FD      AND #$FD      Bit 1 löschen
FAC3   8D 0C 1C   STA $1C0C
FAC6   60         RTS

******************************  Formatierung
FAC7   A5 51      LDA $51       Tracknummer
FAC9   10 2A      BPL $FAF5     Formatierung bereits im Gange ?
FACB   A6 3D      LDX $3D       Drivenummer
FACD   A9 60      LDA #$60      Flag für Kopftransport
FACF   95 20      STA $20,X     setzen
FAD1   A9 01      LDA #$01
FAD3   95 22      STA $22,X     Zieltrack setzen
FAD5   85 51      STA $51       laufende Tracknummer bei der Formatierung
FAD7   A9 A4      LDA #$A4      164
FAD9   85 4A      STA $4A       Schrittzähler für Kopftransport
FADB   AD 00 1C   LDA $1C00
FADE   29 FC      AND #$FC      Steppermotor ein
FAE0   8D 00 1C   STA $1C00
FAE3   A9 0A      LDA #$0A      10
FAE5   8D 20 06   STA $0620     Fehlerzähler
FAE8   A9 A0      LDA #$A0      $6210/$622 = 4000
FAEA   8D 21 06   STA $0621     zur Bestimmung der Trackkapazität initialisieren
```

```
FAED   A9 0F      LDA #$0F      4000 < Kapazität < 2*4000 Bytes
FAEF   8D 22 06   STA $0622
FAF2   4C 9C F9   JMP $F99C     zurück in Jobschleife

FAF5   A0 00      LDY #$00
FAF7   D1 32      CMP ($32),Y
FAF9   F0 05      BEQ $FB00
FAFB   91 32      STA ($32),Y
FAFD   4C 9C F9   JMP $F99C     zur Jobschleife

FB00   AD 00 1C   LDA $1C00
FB03   29 10      AND #$10      'Write Protect' ?
FB05   D0 05      BNE $FB0C     nein
FB07   A9 08      LDA #$08
FB09   4C D3 FD   JMP $FDD3     26, 'write protect on'

FB0C   20 A3 FD   JSR $FDA3     10240 mal Kode $FF auf Diskette schreiben
FB0F   20 C3 FD   JSR $FDC3     ($621/$622) mal Kode $FF auf Diskette
FB12   A9 55      LDA #$55      $55
FB14   8D 01 1C   STA $1C01     zum Schreibkopf
FB17   20 C3 FD   JSR $FDC3     und ($621/$622) mal auf Diskette
FB1A   20 00 FE   JSR $FE00     auf Lesen umschalten
FB1D   20 56 F5   JSR $F556     Timer setzen, $FF (SYNC) suchen
FB20   A9 40      LDA #$40
FB22   0D 0B 18   ORA $180B     Timer 1 free running
FB25   8D 0B 18   STA $180B
FB28   A9 62      LDA #$62      98 Taktzyklen, ca. 0.1 ms
FB2A   8D 06 18   STA $1806
FB2D   A9 00      LDA #$00
FB2F   8D 07 18   STA $1807
FB32   8D 05 18   STA $1805     Timer starten
FB35   A0 00      LDY #$00      Zähler auf Null
FB37   A2 00      LDX #$00
FB39   2C 00 1C   BIT $1C00     SYNC gefunden ?
FB3C   30 FB      BMI $FB39     nein, warten
FB3E   2C 00 1C   BIT $1C00     SYNC gefunden ?
FB41   10 FB      BPL $FB3E     warten bis SYNC-Bereich zu Ende
FB43   AD 04 18   LDA $1804     Interruptflag Timer zurücksetzen
FB46   2C 00 1C   BIT $1C00     SYNC gefunden ?
FB49   10 11      BPL $FB5C     nicht SYNC-Bereich ($55) zu Ende ?
FB4B   AD 0D 18   LDA $180D     Interrupt-Flag-Register
FB4E   0A         ASL  A        Timerflag nach Bit sieben
FB4F   10 F5      BPL $FB46     Timer noch nicht abgelaufen ?
FB51   E8         INX           Zähler erhöhen
FB52   D0 EF      BNE $FB43
FB54   C8         INY           Hi-Byte des Zählers erhöhen
FB55   D0 EC      BNE $FB43
FB57   A9 02      LDA #$02      Überlauf, dann Fehler
FB59   4C D3 FD   JMP $FDD3     20, 'read error'

FB5C   86 71      STX $71       Zählerstand gleich Dauer des $55-Bereichs
FB5E   84 72      STY $72       merken
FB60   A2 00      LDX #$00
FB62   A0 00      LDY #$00      Zähler wieder auf null
FB64   AD 04 18   LDA $1804     Timer 1 Interruptflag rücksetzen
FB67   2C 00 1C   BIT $1C00     SYNC gefunden ?
```

```
FB6A   30 11      BMI $FB7D     ja
FB6C   AD 0D 18   LDA $180D     Interrupt-Flag-Register
FB6F   0A         ASL  A        Timerflag nach Bit 7
FB70   10 F5      BPL $FB67     nein, warten bis Timer abgelaufen
FB72   E8         INX
FB73   D0 EF      BNE $FB64     Zähler erhöhen
FB75   C8         INY
FB76   D0 EC      BNE $FB64
FB78   A9 02      LDA #$02      Überlauf, dann Fehler
FB7A   4C D3 FD   JMP $FDD3     20, 'read error'

FB7D   38         SEC
FB7E   8A         TXA
FB7F   E5 71      SBC $71       Differenz zwischen Zählerstand ($55)
FB81   AA         TAX
FB82   85 70      STA $70
FB84   98         TYA           und Wert für $FF-Bereich
FB85   E5 72      SBC $72
FB87   A8         TAY           nach $70/$71 bringen
FB88   85 71      STA $71
FB8A   10 0B      BPL $FB97     Differenz positiv ?
FB8C   49 FF      EOR #$FF
FB8E   A8         TAY
FB8F   8A         TXA
FB90   49 FF      EOR #$FF      Absolutwert der Differenz berechnen
FB92   AA         TAX
FB93   E8         INX
FB94   D0 01      BNE $FB97
FB96   C8         INY
FB97   98         TYA
FB98   D0 04      BNE $FB9E
FB9A   E0 04      CPX #$04      Differenz kleiner 4 * 0.1 ms ?
FB9C   90 18      BCC $FBB6     ja
FB9E   06 70      ASL $70
FBA0   26 71      ROL $71       Differenz verdoppeln
FBA2   18         CLC
FBA3   A5 70      LDA $70
FBA5   6D 21 06   ADC $0621
FBA8   8D 21 06   STA $0621     zu Ausgangswert 4000 addieren
FBAB   A5 71      LDA $71
FBAD   6D 22 06   ADC $0622
FBB0   8D 22 06   STA $0622
FBB3   4C 0C FB   JMP $FB0C     wiederholen, bis Differenz kleiner 0.4 ms

FBB6   A2 00      LDX #$00
FBB8   A0 00      LDY #$00      Zähler wieder auf null
FBBA   B8         CLV
FBBB   AD 00 1C   LDA $1C00     SYNC ?
FBBE   10 0E      BPL $FBCE     nein
FBC0   50 F9      BVC $FBBB     Byte Ready ?
FBC2   B8         CLV
FBC3   E8         INX
FBC4   D0 F5      BNE $FBBB     Zähler erhöhen
FBC6   C8         INY
FBC7   D0 F2      BNE $FBBB
FBC9   A9 03      LDA #$03      Überlauf, dann Fehler
```

```
FBCB   4C D3 FD   JMP $FDD3     21, read error

FBCE   8A         TXA
FBCF   0A         ASL  A       Zähler verdoppeln
FBD0   8D 25 06   STA $0625
FBD3   98         TYA
FBD4   2A         ROL  A       und nach $624/$625 als Spurkapazität
FBD5   8D 24 06   STA $0624
FBD8   A9 BF      LDA #$BF
FBDA   2D 0B 18   AND $180B
FBDD   8D 0B 18   STA $180B
FBE0   A9 66      LDA #$66     102
FBE2   8D 26 06   STA $0626
FBE5   A6 43      LDX $43      Anzahl der Sektoren in diesem Track
FBE7   A0 00      LDY #$00
FBE9   98         TYA
FBEA   18         CLC
FBEB   6D 26 06   ADC $0626
FBEE   90 01      BCC $FBF1
FBF0   C8         INY
FBF1   C8         INY
FBF2   CA         DEX
FBF3   D0 F5      BNE $FBEA    Berechnung der Anzahl aller Bytes
FBF5   49 FF      EOR #$FF     in den Blockzwischenräumen
FBF7   38         SEC
FBF8   69 00      ADC #$00
FBFA   18         CLC
FBFB   6D 25 06   ADC $0625
FBFE   B0 03      BCS $FC03
FC00   CE 24 06   DEC $0624
FC03   AA         TAX
FC04   98         TYA
FC05   49 FF      EOR #$FF
FC07   38         SEC
FC08   69 00      ADC #$00
FC0A   18         CLC
FC0B   6D 24 06   ADC $0624    Ergebnis in A/X
FC0E   10 05      BPL $FC15
FC10   A9 04      LDA #$04
FC12   4C D3 FD   JMP $FDD3    22, 'read error'

FC15   A8         TAY
FC16   8A         TXA
FC17   A2 00      LDX #$00
FC19   38         SEC          Die Gesamtzahl wird durch die Anzahl
FC1A   E5 43      SBC $43      der Sektoren ($43) dividiert
FC1C   B0 03      BCS $FC21
FC1E   88         DEY
FC1F   30 03      BMI $FC24
FC21   E8         INX
FC22   D0 F5      BNE $FC19
FC24   8E 26 06   STX $0626    Zahl der Bytes pro Zwischenraum
FC27   E0 04      CPX #$04     mit Minimalwert vergleichen
FC29   B0 05      BCS $FC30    ok
FC2B   A9 05      LDA #$05
FC2D   4C D3 FD   JMP $FDD3    23, 'read error'
```

```
FC30  18        CLC           Rest der Division
FC31  65 43     ADC $43       plus Anzahl der Sektoren
FC33  8D 27 06  STA $0627     merken
FC36  A9 00     LDA #$00
FC38  8D 28 06  STA $0628     Zähler für Sektoren
FC3B  A0 00     LDY #$00      Zähler lo
FC3D  A6 3D     LDX $3D       Drivenummer
FC3F  A5 39     LDA $39       Konstante 8, Kennzeichen für Headeranfang
FC41  99 00 03  STA $0300,Y   in Puffer
FC44  C8        INY
FC45  C8        INY
FC46  AD 28 06  LDA $0628     Sektornummer
FC49  99 00 03  STA $0300,Y   in Puffer
FC4C  C8        INY
FC4D  A5 51     LDA $51       Tracknummer
FC4F  99 00 03  STA $0300,Y   in Puffer
FC52  C8        INY
FC53  B5 13     LDA $13,X     ID 2
FC55  99 00 03  STA $0300,Y   in Puffer
FC58  C8        INY
FC59  B5 12     LDA $12,X     ID 1
FC5B  99 00 03  STA $0300,Y   in Puffer
FC5E  C8        INY
FC5F  A9 0F     LDA #$0F      15
FC61  99 00 03  STA $0300,Y   in Puffer
FC64  C8        INY
FC65  99 00 03  STA $0300,Y   15 in Puffer
FC68  C8        INY
FC69  A9 00     LDA #$00
FC6B  59 FA 02  EOR $02FA,Y
FC6E  59 FB 02  EOR $02FB,Y
FC71  59 FC 02  EOR $02FC,Y   Prüfsumme bilden
FC74  59 FD 02  EOR $02FD,Y
FC77  99 F9 02  STA $02F9,Y
FC7A  EE 28 06  INC $0628     Zähler erhöhen
FC7D  AD 28 06  LDA $0628     Zähler
FC80  C5 43     CMP $43       mit Anzahl der Sektoren vergleichen
FC82  90 BB     BCC $FC3F     kleiner, dann weiter machen
FC84  98        TYA
FC85  48        PHA
FC86  E8        INX
FC87  8A        TXA
FC88  9D 00 05  STA $0500,X
FC8B  E8        INX
FC8C  D0 FA     BNE $FC88
FC8E  A9 03     LDA #$03      Pufferzeiger auf $300
FC90  85 31     STA $31
FC92  20 30 FE  JSR $FE30
FC95  68        PLA
FC96  A8        TAY
FC97  88        DEY
FC98  20 E5 FD  JSR $FDE5     Pufferdaten kopieren
FC9B  20 F5 FD  JSR $FDF5     Daten in Puffer kopieren
FC9E  A9 05     LDA #$05
FCA0  85 31     STA $31       Pufferzeiger auf $500
FCA2  20 E9 F5  JSR $F5E9     Parity für Datenpuffer berechnen
```

```
FCA5   85 3A      STA $3A       und merken
FCA7   20 8F F7   JSR $F78F
FCAA   A9 00      LDA #$00
FCAC   85 32      STA $32
FCAE   20 0E FE   JSR $FE0E     Umschalten auf Schreiben 10240 mal $55 schreiben
FCB1   A9 FF      LDA #$FF
FCB3   8D 01 1C   STA $1C01     zum Schreibkopf
FCB6   A2 05      LDX #$05      5 mal  $FF schreiben
FCB8   50 FE      BVC $FCB8     Byte Ready ?
FCBA   B8         CLV
FCBB   CA         DEX
FCBC   D0 FA      BNE $FCB8
FCBE   A2 0A      LDX #$0A      10 mal
FCC0   A4 32      LDY $32       Pufferzeiger
FCC2   50 FE      BVC $FCC2     Byte Ready ?
FCC4   B8         CLV
FCC5   B9 00 03   LDA $0300,Y   Daten aus Puffer
FCC8   8D 01 1C   STA $1C01     schreiben
FCCB   C8         INY
FCCC   CA         DEX           schon 10 Daten geschrieben ?
FCCD   D0 F3      BNE $FCC2
FCCF   A2 09      LDX #$09      9 mal
FCD1   50 FE      BVC $FCD1     Byte Ready ?
FCD3   B8         CLV
FCD4   A9 55      LDA #$55      $55
FCD6   8D 01 1C   STA $1C01     schreiben
FCD9   CA         DEX
FCDA   D0 F5      BNE $FCD1     schon 9 mal ?
FCDC   A9 FF      LDA #$FF      $FF
FCDE   A2 05      LDX #$05      5 mal
FCE0   50 FE      BVC $FCE0     Byte Ready ?
FCE2   B8         CLV
FCE3   8D 01 1C   STA $1C01     zum Schreibkopf
FCE6   CA         DEX
FCE7   D0 F7      BNE $FCE0
FCE9   A2 BB      LDX #$BB
FCEB   50 FE      BVC $FCEB
FCED   B8         CLV
FCEE   BD 00 01   LDA $0100,X   Bereich $1BB bis $1FF
FCF1   8D 01 1C   STA $1C01     schreiben
FCF4   E8         INX
FCF5   D0 F4      BNE $FCEB
FCF7   A0 00      LDY #$00
FCF9   50 FE      BVC $FCF9     Byte Ready ?
FCFB   B8         CLV
FCFC   B1 30      LDA ($30),Y   256 Byte Daten
FCFE   8D 01 1C   STA $1C01     auf Diskette schreiben
FD01   C8         INY
FD02   D0 F5      BNE $FCF9
FD04   A9 55      LDA #$55      $55
FD06   AE 26 06   LDX $0626     ($626) mal
FD09   50 FE      BVC $FD09
FD0B   B8         CLV
FD0C   8D 01 1C   STA $1C01     schreiben
FD0F   CA         DEX
FD10   D0 F7      BNE $FD09
```

```
FD12   A5 32      LDA $32
FD14   18         CLC
FD15   69 0A      ADC #$0A      plus 10
FD17   85 32      STA $32
FD19   CE 28 06   DEC $0628     Sektornummer erniedrigen
FD1C   D0 93      BNE $FCB1
FD1E   50 FE      BVC $FD1E     Byte Ready ?
FD20   B8         CLV
FD21   50 FE      BVC $FD21     Byte Ready ?
FD23   B8         CLV
FD24   20 00 FE   JSR $FE00     Umschalten auf Lesen
FD27   A9 C8      LDA #$C8      200
FD29   8D 23 06   STA $0623
FD2C   A9 00      LDA #$00
FD2E   85 30      STA $30
FD30   A9 03      LDA #$03      Pufferzeiger auf $300
FD32   85 31      STA $31
FD34   A5 43      LDA $43       Anzahl der Sektoren pro Track
FD36   8D 28 06   STA $0628
FD39   20 56 F5   JSR $F556     SYNC abwarten
FD3C   A2 0A      LDX #$0A      10 Daten
FD3E   A0 00      LDY #$00
FD40   50 FE      BVC $FD40     Byte Ready ?
FD42   B8         CLV
FD43   AD 01 1C   LDA $1C01     Byte lesen
FD46   D1 30      CMP ($30),Y   mit Daten im Puffer vergleichen
FD48   D0 0E      BNE $FD58     ungleich, Fehler
FD4A   C8         INY
FD4B   CA         DEX
FD4C   D0 F2      BNE $FD40
FD4E   18         CLC
FD4F   A5 30      LDA $30
FD51   69 0A      ADC #$0A      Zeiger um 10 erhöhen
FD53   85 30      STA $30
FD55   4C 62 FD   JMP $FD62

FD58   CE 23 06   DEC $0623     Zähler für Versuche erniedrigen
FD5B   D0 CF      BNE $FD2C     noch nicht null ?
FD5D   A9 06      LDA #$06      sonst Fehler
FD5F   4C D3 FD   JMP $FDD3     24, 'read error'

FD62   20 56 F5   JSR $F556     SYNC abwarten
FD65   A0 BB      LDY #$BB
FD67   50 FE      BVC $FD67     Byte Ready ?
FD69   B8         CLV
FD6A   AD 01 1C   LDA $1C01     Byte lesen
FD6D   D9 00 01   CMP $0100,Y   und mit Pufferinhalt vergleichen
FD70   D0 E6      BNE $FD58     ungleich, Fehler
FD72   C8         INY
FD73   D0 F2      BNE $FD67     nächstes Byte
FD75   A2 FC      LDX #$FC
FD77   50 FE      BVC $FD77     Byte Ready ?
FD79   B8         CLV
FD7A   AD 01 1C   LDA $1C01     Byte lesen
FD7D   D9 00 05   CMP $0500,Y   mit Pufferinhalt vergleichen
FD80   D0 D6      BNE $FD58     ungleich, dann Fehler
```

```
FD82   C8          INY
FD83   CA          DEX             nächstes Byte
FD84   D0 F1       BNE $FD77
FD86   CE 28 06    DEC $0628       Sektorzähler erniedrigen
FD89   D0 AE       BNE $FD39       noch nicht null ?
FD8B   E6 51       INC $51         Tracknummer erhöhen
FD8D   A5 51       LDA $51
FD8F   C9 24       CMP #$24        mit 36, höchster Tracknummer +1 vergleichen
FD91   B0 03       BCS $FD96       größer, dann Formatierung fertig
FD93   4C 9C F9    JMP $F99C       weiter machen

FD96   A9 FF       LDA #$FF
FD98   85 51       STA $51         Tracknummer auf $FF
FD9A   A9 00       LDA #$00
FD9C   85 50       STA $50
FD9E   A9 01       LDA #$01
FDA0   4C 69 F9    JMP $F969       ok

****************************** 10240 mal $FF schreiben
FDA3   AD 0C 1C    LDA $1C0C
FDA6   29 1F       AND #$1F        PCR auf Schreiben umschalten
FDA8   09 C0       ORA #$C0
FDAA   8D 0C 1C    STA $1C0C
FDAD   A9 FF       LDA #$FF
FDAF   8D 03 1C    STA $1C03       Port A (Schreib/Lesekopf) auf Ausgabe
FDB2   8D 01 1C    STA $1C01       $FF auf Diskette schreiben
FDB5   A2 28       LDX #$28        40
FDB7   A0 00       LDY #$00
FDB9   50 FE       BVC $FDB9       Byte Ready ?
FDBB   B8          CLV
FDBC   88          DEY
FDBD   D0 FA       BNE $FDB9
FDBF   CA          DEX
FDC0   D0 F7       BNE $FDB9
FDC2   60          RTS

****************************** ($621/$622) mal schreiben/lesen
FDC3   AE 21 06    LDX $0621
FDC6   AC 22 06    LDY $0622
FDC9   50 FE       BVC $FDC9       Byte Ready ?
FDCB   B8          CLV
FDCC   CA          DEX
FDCD   D0 FA       BNE $FDC9
FDCF   88          DEY
FDD0   10 F7       BPL $FDC9
FDD2   60          RTS

****************************** Versuchezähler beim Formatieren
FDD3   CE 20 06    DEC $0620       Anzahl der Versuche erniedrigen
FDD6   F0 03       BEQ $FDDB       null, dann Fehler melden
FDD8   4C 9C F9    JMP $F99C       weiter machen

FDDB   A0 FF       LDY #$FF
FDDD   84 51       STY $51         Flag für Formatierung beendet
FDDF   C8          INY
FDE0   84 50       STY $50
```

```
FDE2   4C 69 F9   JMP $F969      Fehlerabschluß

******************************
FDE5   B9 00 03   LDA $0300,Y
FDE8   99 45 03   STA $0345,Y
FDEB   88         DEY            Pufferinhalt kopieren
FDEC   D0 F7      BNE $FDE5
FDEE   AD 00 03   LDA $0300
FDF1   8D 45 03   STA $0345
FDF4   60         RTS

******************************
FDF5   A0 44      LDY #$44
FDF7   B9 BB 01   LDA $01BB,Y   $1BB bis $1FF
FDFA   91 30      STA ($30),Y   in Puffer $30/$31 schreiben
FDFC   88         DEY
FDFD   10 F8      BPL $FDF7
FDFF   60         RTS

******************************  Umschalten auf Lesen
FE00   AD 0C 1C   LDA $1C0C
FE03   09 E0      ORA #$E0      PCR auf Lesen umschalten
FE05   8D 0C 1C   STA $1C0C
FE08   A9 00      LDA #$00
FE0A   8D 03 1C   STA $1C03     Port A auf Eingang
FE0D   60         RTS

******************************  10240 mal $55 schreiben
FE0E   AD 0C 1C   LDA $1C0C
FE11   29 1F      AND #$1F
FE13   09 C0      ORA #$C0      PCR auf Schreiben umschalten
FE15   8D 0C 1C   STA $1C0C
FE18   A9 FF      LDA #$FF
FE1A   8D 03 1C   STA $1C03     Port A auf Ausgabe zum Schreibkopf
FE1D   A9 55      LDA #$55      %01010101
FE1F   8D 01 1C   STA $1C01     auf Port A zum Schreibkopf
FE22   A2 28      LDX #$28
FE24   A0 00      LDY #$00
FE26   50 FE      BVC $FE26     Byte Ready von Schreibelektronik ?
FE28   B8         CLV
FE29   88         DEY
FE2A   D0 FA      BNE $FE26     10240 mal
FE2C   CA         DEX
FE2D   D0 F7      BNE $FE26
FE2F   60         RTS

******************************
FE30   A9 00      LDA #$00
FE32   85 30      STA $30
FE34   85 2E      STA $2E
FE36   85 36      STA $36
FE38   A9 BB      LDA #$BB
FE3A   85 34      STA $34
FE3C   A5 31      LDA $31
FE3E   85 2F      STA $2F
FE40   A9 01      LDA #$01
```

```
FE42   85 31      STA $31
FE44   A4 36      LDY $36
FE46   B1 2E      LDA ($2E),Y
FE48   85 52      STA $52
FE4A   C8         INY
FE4B   B1 2E      LDA ($2E),Y
FE4D   85 53      STA $53
FE4F   C8         INY
FE50   B1 2E      LDA ($2E),Y
FE52   85 54      STA $54
FE54   C8         INY
FE55   B1 2E      LDA ($2E),Y
FE57   85 55      STA $55
FE59   C8         INY
FE5A   F0 08      BEQ $FE64
FE5C   84 36      STY $36
FE5E   20 D0 F6   JSR $F6D0
FE61   4C 44 FE   JMP $FE44

FE64   4C D0 F6   JMP $F6D0

******************************  Interrupt-Routine
FE67   48         PHA
FE68   8A         TXA
FE69   48         PHA           Register retten
FE6A   98         TYA
FE6B   48         PHA
FE6C   AD 0D 18   LDA $180D     Interrupt vom seriellen Bus (ATN IN) ?
FE6F   29 02      AND #$02
FE71   F0 03      BEQ $FE76     nein
FE73   20 53 E8   JSR $E853     seriellen Bus bedienen
FE76   AD 0D 1C   LDA $1C0D     Interrupt von Timer 1 ?
FE79   0A         ASL  A
FE7A   10 03      BPL $FE7F     nein
FE7C   20 B0 F2   JSR $F2B0     IRQ-Routine für Disk-Controller
FE7F   68         PLA
FE80   A8         TAY
FE81   68         PLA           Register zurückholen
FE82   AA         TAX
FE83   68         PLA
FE84   40         RTI

******************************  Konstanten für Diskettenformat
FE85 12                         18, Track für BAM und Directory
FE86 04                         Start der BAM ab Position 4
FE87 04                         4 Bytes in BAM für jeden Track
FE88 90                         $90 = 144, Ende BAM, Start Diskname

****************************** Tabelle der Kommandoworte
FE89 56 49 44 4D 42 55          'V', 'I', 'D', 'M', 'B', 'U'
FE8F 50 26 43 52 53 4E          'P', '&', 'C', 'R', 'S', 'N'

****************************** Lo-Byte der Adressen der Befehle
FE95 84 05 C1 F8 1B 5C
FE9F 07 A3 F0 88 23 0D
```

```
****************************** Hi-Byte der Adressen der Befehle
FEA1 ED D0 C8 CA CC CB
FEA7 E2 E7 C8 CA C8 EE

******************************
FEAD 51 DD 1C 9E 1C             Bytes für Syntaxprüfung

******************************  File-Betriebsarten
FEB2 52 57 41 4D                'R', 'W', 'A', 'M'

******************************  Filetypen
FEB6 44 53 50 55 4C             'D', 'S', 'P', 'U', 'L'

******************************  Namen der Filetypen
FEBB 44 53 50 55 52      1. Buchstabe des Filetyps 'D', 'S', 'P', 'U', 'R'
FEC0 45 45 52 53 45      2. Buchstabe       "      'E', 'E', 'R', 'S', 'E'
FEC5 4C 51 47 52 4C      3. Buchstabe       "      'L', 'Q', 'G', 'R', 'L'

******************************
FECA 08 00 00

******************************
FECD 3F 7F BF FF                Masken für Bit-Befehl

******************************  Anzahl der Sektoren pro Track
FED1 11 12 13 15                17, 18, 19, 21

******************************  Konstanten für Diskettenformat
FED5 41                         'A' Kennzeichen für 1541-Format
FED6 04                         4 Tracknummern
FED7 24                         36, höchste Tracknummer + 1
FED8 1F 19 12                   31, 25, 18 Tracks mit Wechsel Anzahl Sektoren

******************************
FEDB 01 FF FF 01 00             Steuerbytes für Kopfpositionierung

******************************  Adressen der Pufferspeicher
FEE0 03 04 05 06 07             High-Bytes

******************************
FEE5 07 0E

******************************  Vom UI-Befehl
FEE7   6C 65 00   JMP ($0065)

******************************  von der Diagnose-Routine
FEEA   8D 00 1C   STA $1C00     LED einschalten
FEED   8D 02 1C   STA $1C02     Port auf Ausgabe
FEF0   4C 7D EA   JMP $EA7D     zurück zur Diagnose-Routine

******************************  Verzögerungsschleife für seriellen Bus
FEF3   8A         TXA
FEF4   A2 05      LDX #$05
FEF6   CA         DEX           ca. 40 Mikrosekunden
FEF7   D0 FD      BNE $FEF6
FEF9   AA         TAX
```

```
FEFA   60         RTS

******************************  Datenausgabe auf seriellen Bus
FEFB   20 AE E9   JSR $E9AE     CLOCK OUT hi
FEFE   4C 9C E9   JMP $E99C     DATA OUT lo

******************************  UI-Vektor
FF01   AD 02 02   LDA $0202
FF04   C9 2D      CMP #$2D      '-'
FF06   F0 05      BEQ $FF0D
FF08   38         SEC
FF09   E9 2B      SBC #$2B      '+'
FF0B   D0 DA      BNE $FEE7     indirekter Sprung über ($65)
FF0D   85 23      STA $23
FF0F   60         RTS

******************************
FF10 AA ...
FFE1 ... AA

******************************
FFE2 52 53 52 AA
FFE6 C6 C8 8F F9

*********************  USER-Vektoren
FFEA 5F CD            UA, U1, $CD5F
FFEC 97 CD            UB, U2, $CD97
FFEE 00 05            UC, U3, $0500
FFF0 03 05            UD, U4, $0503
FFF2 06 05            UE, U5, $0506
FFF4 09 05            UF, U6, $0509
FFF6 0C 05            UG, U7, $050C
FFF8 0F 05            UH, U8, $050F
FFFA 01 FF            UI, U9, $FF01   (NMI-Vektor wird nicht benutzt)

*********************  Hardware-Vektoren
FFFC A0 EA            $EAA0   RESET-  und UJ- bzw. U:- Vektor
FFFE 67 FE            $FE67   IRQ-Vektor
```

4.1 Dienstprogramme

4.1.1 Anzeige sämtlicher Fileparameter

Dem Directory sind nicht alle Informationen eines Files zu entnehmen. Vieleicht standen Sie auch einmal vor dem Problem, daß Sie z.B. die Anfangsadresse eines auf Diskette abgelegten Programms benötigen. Dann kennen Sie sicher auch die Umstände, die mit der Ermittlung dieser Anfangsadresse verbunden sind.
Ein weiteres Beispiel ist die Recordlänge eines relativen Files. Sie kann nur mit großem Programmieraufwand ermittelt werden, wenn sie in Vergessenheit geraten ist.
Dies sind nur zwei der vielen Fileparameter, die mit dem folgendem Programm äußerst einfach ermittelt und angezeigt werden können. Die Fileparameter sind natürlich auch vom Filetyp abhängig. So kann z.B. einem relativen File keine Anfangsadresse zugeordnet werden. Die folgende Tabelle stellt die mit diesem Programm ermittelbaren Parameter der einzelnen Filetypen dar:

PARAMETER	FILETYP				
	DEL	SEQ	PRG	USR	REL
File geschlossen?	X	X	X	X	X
File geschützt?	X	X	X	X	X
belegte Blöcke	X	X	X	X	X
Recordlänge					X
Side-Sector-Blöcke					X
Datenblöcke					X
Records					X
Anfangsadresse			X		
freie Blöcke Disk	X	X	X	X	X
belegte Bl. Disk	X	X	X	X	X

Um einen guten Überblick über die Arbeitsweise dieses Programms zu erhalten, was wohl im Interesse jedes ernsthaften Programmierers liegt, ist es bis ins letzte Detail dokumentiert. Einer Übersicht der im Programm verwendeten Variablen folgt eine zeilenorientierte Dokumentation:

Im Programm verwendete Variablen:

numerische Variablen

T - Track (Spur) des aktuellen Blocks der Fileeinträge im Directory

S - Sektor des aktuellen Blocks der Fileeinträge im Directory
FL - Flag, das gesetzt wird, wenn die von der Diskette gelesenen Filenamen zum Auflisten, nicht zum Vergleichen mit dem gesuchten File benutzt werden
TY - Filetyp des angegebenen Files (Byte 0 des Eintrags)
FT - Halbbyte des Filetyps (Bit 0 bis 3), enthält den eigentlichen Filetyp
LB - Low-Byte einer von der Diskette gelesenen Anfangsadresse
HB - High-Byte einer von der Diskette gelesenen Anfangsadresse
BL - Anzahl der vom File belegten Blöcke
RL - Recordlänge eines realtiven Files
DT - Track (Spur) des ersten Datenblocks eines Program-Files, der die Anfangsadresse enthält
DS - Sektor des ersten Datenblocks eines Program-Files
AA - Anfangsadresse eines Program-Files
BF - Anzahl der freien Blöcke auf der Diskette
BB - Anzahl der belegten Blöcke auf der Diskette
BS - Anzahl der Side-Sector-Blöcke in einem relativen File
RC - Anzahl der Records in einem relativen File

Stringvariablen

F$ - Name des gesuchten Files
FF$ - Enthält den aktuellen Filenamen aus der Directory
FT$ - Filetyp (Klartext)
GE$ - Konstante, die angibt, ob das File geschlosssen ist (enthält "JA" oder "NEIN")
SA$ - Konstante, die bestimmt, ob das File geschützt ist (enthält "JA" oder "NEIN")
RE$ - enthält CHR$(18), REVERSE ON
RA$ - enthält CHR$(146), REVERSE OFF

Dokumentation des Programms:

110	Setzt Farbcode des Bildschirms
120 - 200	Programmkopf
210 - 230	Abfrage, ob Namen aufgelistet werden sollen. Setzt Flag FL auf 1 und führt Routine 280 - 490 aus.
250 - 270	Eingabe des Filenamens. Fordert erneute Eingabe, wenn Filename größer als 16 Zeichen
280 - 490	liest die Filenamen aus der Directory und gibt Sie entwerder aus (FL=1) oder vergleicht sie mit dem gesuchten Filenamen
500 - 530	liest das Byte 0 (Filetyp) des Fileeintrags des gesuchten Files und speichert es in TY. Zusätzlich wird das rechte Halbbyte in FT gespeichert
540 - 590	prüft den Filetyp und speichert dessen Klartext in FT$, prüft auf ungültigem Filetyp

600 - 610 prüft das Bit 7 des Filetyp-Bytes (File geschlossen?) und speichert das Resultat in GE$

620 - 630 prüft das Bit 6 des Filetyp-Bytes (File geschützt?) und speichert das Resultat in SA$

640 - 690 liest die Anzahl der vom File belegten Blöcke aus den Bytes 28 und 29 des Eintrags und speichert sie in BL

700 - 730 falls ein relatives File vorliegt, wird hier die Recordlänge aus Byte 21 des Eintrags gelesen und nach RL gebracht

740 - 880 falls ein Program-File vorliegt, wird die Anfangsadresse des Files aus seinem ersten Datenblock ermittelt und in AA abgelegt

890 - 980 berechnet die freien Blöcke der Diskette, indem das jeweils erste Byte des spurkennzeichnenden BAM-Ausschnittes gelesen und in BF aufaddiert wird. Die belegten Blöcke werden dann mit BB = 644 - BF ermittelt

990 - 1020 hier wird bei relativen Files mit Hilfe der Recordlänge (RL) und der vom vom File belegten Blöcke die Anzahl der Side-Sector-Blöcke (BS) und die Anzahl der Records (RC) errechnet. Da für jeweils 120 Blöcke eines relativen Files ein Side-Sector-Block gebildet wird, wird die Anzahl der Side-Sector-Blöcke mit BS = BL / 121 und der Aufrundung auf die nächste ganze Zahl berechnet. Die restlichen Blöcke multipliziert mit 254 und dividiert mit der Recordlänge ergeben die Anzahl der Records im File.

1040 - 1230 hier werden die ermittelten Daten wahlweise auf dem Bildschirm oder auf dem Drucker ausgegeben. Die Fileparameter werden REVERSE angezeigt.

1240 - 1280 ermöglicht die Parameterausgabe eines weiteren Files

Das Programm wurde auf einem CBM 64 erstellt. Trotzdem ist es ohne großartige Änderungen auf dem VC 20 lauffähig. Lediglich die Zeile 110, wo die Bildschirmfarbe gesetzt wird, muß dem VC 20 angepasst werden.

BASIC-Listing des Programms:

```
100 CLR
110 POKE53280,2:POKE53281,2:PRINTCHR$(158);CHR$(147);
120 PRINTTAB(6);"============================"
130 PRINTTAB(6);"ANZEIGE ALLER FILEPARAMETER"
140 PRINTTAB(6);"============================"
150 PRINT:PRINT
160 PRINT"MIT DIESEM  PROGRAMM KOENNEN SAEMTLICHE"
170 PRINT"PARAMETER  EINES  FILES  WAHLWEISE  AUF"
180 PRINT"BILDSCHIRM ODER DRUCKER AUSGEGEBEN WER-"
190 PRINT"DEN."
200 PRINT:PRINT
210 PRINT"FILENAMEN AUFLISTEN (J/N)?"
220 GETX$:IFX$=""ORX$<>"J"ANDX$<>"N"THEN220
230 IFX$="J"THENFL=1:GOSUB280
```

```
240 FL=0
250 INPUT"NAME DES FILES: ";F$
260 IFLEN(F$)<=16THEN280
270 PRINT"FILENAME ZU LANG!":GOTO250
280 OPEN15,8,15,"I0":OPEN2,8,2,"#"
290 T=18:S=1
300 PRINT#15,"U1";2;0;T;S
310 PRINT#15,"B-P";2;0
320 GET#2,X$:IFX$=""THENX$=CHR$(0)
325 T=ASC(X$)
330 GET#2,X$:IFX$=""THENX$=CHR$(0)
340 S=ASC(X$)
350 FORX=0TO7
360 PRINT#15,"B-P";2;X*32+5
370 FF$=""
380 FORY=0TO15
390 GET#2,X$:IFX$=""THENX$=CHR$(0)
400 IFASC(X$)=160THEN430
410 FF$=FF$+X$
420 NEXT Y
430 IFF$=FF$THEN490
440 IFFLTHENPRINT FF$
450 NEXT X
460 IF T=0 THEN480
470 GOTO300
480 CLOSE2:CLOSE15
485 IFFL=0THENPRINT"FILENAME NICHT GEFUNDEN!":GOTO210
490 IFFL THEN RETURN
500 PRINT#15,"B-P";2;X*32+2
510 GET#2,X$:IFX$=""THENX$=CHR$(0)
520 TY=ASC(X$)
530 FT=TYAND15
540 IFFT=0THENFT$="DELETED"
550 IFFT=1THENFT$="SEQUENTIAL"
560 IFFT=2THENFT$="PROGRAM"
570 IFFT=3THENFT$="USER"
580 IFFT=4THENFT$="RELATIVE"
590 IFFT>4THENPRINT"UNGUELTIGER FILETYP!":GOTO200
600 IFTYAND128THENGE$="JA":GOTO620
610 GE$="NEIN"
620 IFTYAND64THENSA$="JA":GOTO640
630 SA$="NEIN"
640 PRINT#15,"B-P";2,X*32+30
650 GET#2,X$:IFX$=""THENX$=CHR$(0)
660 LB=ASC(X$)
670 GET#2,X$:IFX$=""THENX$=CHR$(0)
680 HB=ASC(X$)*256
690 BL=LB+HB
700 IFFT<>4THEN740
710 PRINT#15,"B-P";2;X*32+23
720 GET#2,X$:IFX$=""THENX$=CHR$(0)
730 RL=ASC(X$)
740 IFFT<>2THEN890
750 PRINT#15,"B-P";2;X*32+3
760 GET#2,X$:IFX$=""THENX$=CHR$(0)
770 DT=ASC(X$)
```

```
780 GET#2,X$:IFX$=""THENX$=CHR$(0)
790 DS=ASC(X$)
800 OPEN3,8,3,"#"
810 PRINT#15,"U1";3;0;DT;DS
820 PRINT#15,"B-P";3;2
830 GET#3,X$:IFX$=""THENX$=CHR$(0)
840 LB=ASC(X$)
850 GET#3,X$:IFX$=""THENX$=CHR$(0)
860 HB=ASC(X$)*256
870 AA=LB+HB
880 CLOSE3
890 PRINT#15,"U1";2;0;18;0
900 BF=0
910 FORI=4TO140STEP4
920 IFI=72THEN960
930 PRINT#15,"B-P";2;I
940 GET#2,X$:IFX$=""THENX$=CHR$(0)
950 BF=ASC(X$)+BF
960 NEXT
980 BB=644-BF
990 IFFT<>4THEN1040
1010 BS=BL/121:IFBS<>INT(BS)THENBS=INT(BS+1)
1020 RC=INT(((BL-BS)*254)/RL)
1040 PRINTCHR$(147);"BILDSCHIRM ODER DRUCKER (B/D)?"
1050 GETX$:IFX$=""ORX$<>"B"ANDX$<>"D"THEN1050
1060 RE$=CHR$(18):RA$=CHR$(146)
1070 IFX$="B"THENOPEN1,3:PRINT#1,CHR$(147)
1080 IFX$="D"THENOPEN1,4
1090 PRINT#1,"PARAMETER DES FILES    ";RE$;F$;RO$
1100 PRINT#1,"--------------------------------------"
1110 PRINT#1,"FILETYP:               ";RE$;FT$;RA$:PRINT#1
1120 PRINT#1,"FILE GESCHLOSSEN:      ";RE$;GE$;RA$:PRINT#1
1130 PRINT#1,"FILE GESCHUETZT:       ";RE$;SA$;RA$:PRINT#1
1140 PRINT#1,"BELEGTE BLOECKE:       ";RE$;BL;RA$:PRINT#1
1150 IFFT<>4THEN1200
1160 PRINT#1,"RECORDLAENGE:          ";RE$;RL;RA$:PRINT#1
1170 PRINT#1,"SIDE-SECTOR BLOECKE:   ";RE$;BS;RA$:PRINT#1
1180 PRINT#1,"DATENBLOECKE:          ";RE$;BL-BS;RA$:PRINT#1
1190 PRINT#1,"RECORDS:               ";RE$;RC;RA$:PRINT#1
1200 IFFT=2THEN PRINT#1,"ANFANGSADRESSE:        ";RE$;AA;RA$
     PRINT#1
1210 PRINT#1,"FREIE BLOECKE (DISK):  ";RE$;BF;RA$:PRINT#1
1220 PRINT#1,"BELEGTE BLOECKE (DISK):";RE$;BB;RA$:PRINT#1
1230 CLOSE1
1240 PRINT"WEITER (J/N)?"
1250 CLOSE2:CLOSE15
1260 GETX$:IFX$=""ORX$<>"J"AND X$<>"N"THEN1260
1270 IF X$="J"THEN100
```

4.1.2 Scratch-Schutz von Files - Fileprotect

Wie bereits erwähnt, besteht die Möglichkeit Files auf der VC 1541-Diskette zu schützen und sie auch im Directory als geschützt auszuweisen. Im Byte 0 des Fileeintrags ist der Filetyp enthalten. Das Bit 6, also das Bit mit der dezimalen Wertigkeit 64 kennzeichnet ein geschütztes File. Ist dieses Bit 1, so kann das File nicht mehr mit dem Befehl 'SCRATCH' gelöscht werden. Da das DOS aber keinen Befehl zum Setzen dieses Bits beinhaltet, ist dazu eine BASIC-Befehlsfolge erforderlich.
Mit dem folgenden Programm können Sie:

* alle Files der eingelegten Diskette auf dem Bildschirm anzeigen,
* Files schützen
* Files freigeben
* Files löschen

Es können sowohl ungeschützte als auch geschützte Files gelöscht werden. Bei geschützten Files muß der Wunsch des Löschens zusätzlich bestätigt werden.
Auch dieses Programm ist mit einer Variablentabelle und einer zeilenorientierten Beschreibung ausreichend dokumentiert, sodaß Sie die eine oder andere Befehlsfolge auch in Ihren eigenen Programmen verwenden können.

Liste der Variablen:

GF - Flag, daß in der Routine "lesen/suchen von Files" gesetzt wird, falls der gesuchte Filename gefunden wird
FL - wird gesetzt, wenn die Routine "lesen/suchen von Files nur zum Auflisten aller Files benutzt wird
FT - Variable zur Speicherung des Filetyps
T - Track (Spur) des aktuellen Blocks der Fileeinträge
S - Sektor des aktuellen Blocks der Fileeinträge
TT - Track, in dem sich der Fileeintragsblock des gesuchten Files befindet
SS - Sektor, in dem sich der Fileeintragsblock des gesuchten Files befindet
FF$ - zuletzt gelesener Filename aus der Directory
F$ - eingegebener, gesuchter Filename

Dokumentation des Programms:

100	Setzen der Bildschirmfarbe
110 - 230	Programmkopf und Auswahlmenü
240 - 260	Lesen der Menüauswahl und Aufruf des entsprechenden Unterprogramms
270	Zurück zum Auswahlmenü
280 - 350	Unterprogramm "auflisten aller Files"
310	Bildschirm löschen
320	setzen Flag FL zum Auflisten der Files im

```
                Unterprogramm "lesen/suchen von Files"
      350       Zurücksetzen des Flags und Rücksprung
360 - 600       Unterprogramm "schützen von Files"
      390       Aufrufen  Unterprogramm "Eingabe des Filena-
                mens
      400       Aufrufen  des  Unterprogramms  "lesen/suchen
                von Files"
      410-450   Mit  Hilfe  von  Flag GF testen, ob Filename
                gefunden wurde
      460-480   lesen Filetyp und speichern in FT
      490-500   Testen, ob Files bereits geschützt ist
      510       File schützen (Bit 6 auf 1)
      520-550   Übertragen  des  Filetyps  in den Buffer und
                schreiben des Blocks auf Diskette
      560       Schließen der Kanäle
      570-600   Meldung "File geschützt" und Rücksprung
610 - 850       Unterprogramm "Freigeben von Files"
      640       Aufrufen  Unterprogramm "Eingabe des Filena-
                mens
      650       Aufrufen  Unterprogramm  "lesen /suchen  von
                Files"
      660-700   Testen, ob Filename gefunden wurde
      710-730   Filetyp lesen und in FT speichern
      740-750   Testen, ob File bereits freigegeben ist
      760       Freigeben des Files (Bit 6 auf 0)
      770-800   Übertragen  des  Filetyps  in den Buffer und
                schreiben des Blocks auf Diskette
      810       Schließen der Files
      820-850   Beenden des Unterprogramms
860 - 1170      Unterprogramm "löschen eines Files"
      890       Aufrufen  Unterprogramm "Eingabe des Filena-
                mens
      900       Aufrufen  des  Unterprogramms  "lesen/suchen
                von Files"
      910-950   Testen, ob Filename gefunden wurde
      960-980   Lesen des Filetyps und speichern in FT
      990       Testen, ob File geschützt
      1000-1030 Hinweis,  daß  File  geschützt ist,  mit der
                Möglichkeit, trotzdem zu löschen
      1040-1060 Frage, ob File wirklich gelöscht werden soll
      1070      Bit 6 zurücksetzen, wenn geschützt
      1080-1110 Übertragen  des  Filetyps  in den Buffer und
                schreiben des Blocks auf Diskette
      1120      Initialisieren der Diskette
      1130      Löschen des Files
      1140-1170 Beenden des Unterprogramms
1190 -1560      Unterprogramm "lesen/suchen von Files"
      1220      Öffnen des Befehls- und Datenkanals
      1230-1240 Directory lesen und Bufferpointer setzen
      1250-1320 Testen, ob die Diskette einen  Schreibschutz
                enthält.  Dazu wird die Directory wieder un-
                verändert  zurückgeschrieben  (Zeile  1250).
                Befindet  sich  ein  Schreibschutz  auf  der
                Diskette, so wird die Fehlermeldung 26,WRITE
                PROTECT ON gesetzt.
      1330      Anfangswerte der Variablen für Spur und Sek-
```

tor setzen

1340-1350 Lesen des Fileeintrag-Blocks und Positionieren des Buffer-Pointers auf das erste Byte

1360-1390 Lesen der Adresse des nächsten Fileeintragblocks

1400-1530 Schleife zum Lesen der Filenamen. Die Namen werden dann, je nach Inhalt von Flag FL, entweder auf dem Bildschirm aufgelistet, oder mit dem gesuchten Filenamen verglichen

1540-1560 Wenn die Variable T (Track) eine Null enthält, so folgt kein weiterer Fileeintrag-Block und das Unterprogramm wird beendet

BASIC-Listing des Programms:

```
100 POKE53280,2:POKE53281,2:PRINTCHR$(158);CHR$(147);
110 PRINTTAB(4);"================================"
120 PRINTTAB(4);"LOESCHEN UND SCHUETZEN VON FILES"
130 PRINTTAB(4);"================================"
140 PRINT:PRINT
150 PRINT"MIT DIESEM  PROGRAMM KOENNEN FILES  GE-"
160 PRINT"SCHUETZT,  GELOESCHT  UND  FREIGEGEBEN "
170 PRINT"WERDEN."
180 PRINT:PRINT
190 PRINTTAB(6);" -1- AUFLISTEN ALLER FILES":PRINT
200 PRINTTAB(6);" -2- SCHUETZEN EINES FILES":PRINT
210 PRINTTAB(6);" -3- FREIGEBEN EINES FILES":PRINT
220 PRINTTAB(6);" -4- LOESCHEN  EINES FILES":PRINT
230 PRINTTAB(6);" -5- BEENDEN DES PROGRAMMS":PRINT
240 GETX$:IFX$=""ORVAL(X$)<1ORVAL(X$)>5THEN240
250 IFVAL(X$)=5THENEND
260 ONVAL(X$)GOSUB280,360,610,860
270 GOTO100
280 REM ---------------------
290 REM AUFLISTEN ALLER FILES
300 REM ---------------------
310 PRINTCHR$(147)
320 FL=1:GOSUB1190
330 PRINT:PRINT"WEITER MIT RETURN"
340 INPUTX$
350 FL=0:RETURN
360 REM ---------------------
370 REM SCHUETZEN EINES FILES
380 REM ---------------------
390 GOSUB1580
400 GOSUB1190
410 IFGF= 1 THEN460
420 PRINT"FILE NICHT GEFUNDEN!":PRINT
430 PRINT"WEITER MIT RETURN!"
440 INPUTX$:CLOSE2:CLOSE15
450 RETURN
460 PRINT#15,"B-P";2;X*32+2
470 GET#2,X$:IFX$=""THENX$=CHR$(0)
480 FT=ASC(X$)
490 IF(FT AND 64)=0THEN510
500 PRINT"FILE IST BEREITS GESCHUETZT!":PRINT:GOTO430
```

```
510 FT=(FT OR 64)
520 PRINT#15,"B-P";2;X*32+2
530 PRINT#2,CHR$(FT);
540 PRINT#15,"B-P";2;0
550 PRINT#15,"U2";2;0;TT;SS
560 CLOSE2:CLOSE15
570 PRINT"FILE GESCHUETZT!"
580 PRINT"WEITER MIT RETURN!"
590 INPUTX$
600 CLOSE2:CLOSE15:RETURN
610 REM ---------------------
620 REM FREIGEBEN EINES FILES
630 REM ---------------------
640 GOSUB1580
650 GOSUB1190
660 IFGF= 1 THEN710
670 PRINT"FILE NICHT GEFUNDEN!":PRINT
680 PRINT"WEITER MIT RETURN!"
690 INPUTX$:CLOSE2:CLOSE15
700 RETURN
710 PRINT#15,"B-P";2;X*32+2
720 GET#2,X$:IFX$=""THENX$=CHR$(0)
730 FT=ASC(X$)
740 IF(FT AND 64)=64 THEN760
750 PRINT"FILE IST BEREITS FREIGEGEBEN!":PRINT:GOTO680
760 FT=(FT AND 255-64)
770 PRINT#15,"B-P";2;X*32+2
780 PRINT#2,CHR$(FT);
790 PRINT#15,"B-P";2;0
800 PRINT#15,"U2";2;0;TT;SS
810 CLOSE2:CLOSE15
820 PRINT"FILE FREIGEGEBEN!"
830 PRINT"WEITER MIT RETURN!"
840 INPUTX$
850 RETURN
860 REM ---------------------
870 REM LOESCHEN  EINES FILES
880 REM ---------------------
890 GOSUB1580
900 GOSUB1190
910 IFGF= 1 THEN960
920 PRINT"FILE NICHT GEFUNDEN!":PRINT
930 PRINT"WEITER MIT RETURN!"
940 INPUTX$:CLOSE2:CLOSE15
950 RETURN
960 PRINT#15,"B-P";2;X*32+2
970 GET#2,X$:IFX$=""THENX$=CHR$(0)
980 FT=ASC(X$)
990 IF(FT AND 64)=0THEN1040
1000 PRINT"ACHTUNG! FILE IST GESCHUETZT!"
1010 PRINT"FREIGEBEN UND LOESCHEN (J/N)?"
1020 GETX$:IFX$=""ORX$<>"N"ANDX$<>"J"THEN1020
1030 IFX$="N"THEN1170
1040 PRINT"SICHER (J/N)?"
1050 GETX$:IFX$=""ORX$<>"N"ANDX$<>"J"THEN1050
1060 IFX$="N"THEN1170
```

```
1070 FT=(FT AND 255-64)
1080 PRINT#15,"B-P";2;X*32+2
1090 PRINT#2,CHR$(FT);
1100 PRINT#15,"B-P";2;0
1110 PRINT#15,"U2";2;0;TT;SS
1120 PRINT#15,"I0"
1130 PRINT#15,"S:"+F$
1140 PRINT"FILE GELOESCHT!"
1150 PRINT"WEITER MIT RETURN!"
1160 INPUTX$
1170 CLOSE2:CLOSE15:RETURN
1180 REM
1190 REM ------------------------
1200 REM LESEN / SUCHEN VON FILES
1210 REM ------------------------
1220 OPEN15,8,15,"I0":OPEN2,8,2,"#"
1230 PRINT#15,"U1";2;0;18;0
1240 PRINT#15,"B-P";2;0
1250 PRINT#15,"U2";2;0;18;0
1260 INPUT#15,X1$
1270 IFVAL(X1$)<>26THEN1330
1280 PRINT"BITTE VOR BENUTZUNG DIESES PROGRAMMS DEN";
1290 PRINT"SCHREIBSCHUTZ ENTFERNEN!"
1300 PRINT"WEITER MIT RETURN!"
1310 INPUTX$
1320 CLOSE2:CLOSE15:RETURN
1330 T=18:S=1:TT=18:SS=1
1340 PRINT#15,"U1";2;0;T;S
1345 TT=T:SS=S
1350 PRINT#15,"B-P";2;0
1360 GET#2,X$:IFX$=""THENX$=CHR$(0)
1370 T=ASC(X$)
1380 GET#2,X$:IFX$=""THENX$=CHR$(0)
1390 S=ASC(X$)
1400 FORX=0TO7
1410 PRINT#15,"B-P";2;X*32+2
1420 GET#2,X$:IFX$=""THENX$=CHR$(0)
1430 IFASC(X$)=0THEN1530
1440 PRINT#15,"B-P";2;X*32+5
1450 FF$=""
1460 FORY=0TO15
1470 GET#2,X$:IFX$=""THENX$=CHR$(0)
1480 IFASC(X$)=160THEN 1500
1490 FF$=FF$+X$
1500 NEXTY
1510 IFFLTHENPRINTFF$:GOTO1530
1520 IFF$=FF$THENGF=1:GOTO1570
1530 NEXTX
1540 IFT<>0THEN1340
1550 CLOSE2:CLOSE15
1560 IF FL=0THENPRINT"FILENAME  NICHT GEFUNDEN!"
:FORI=1TO2000:NEXT
1570 RETURN
1580 REM ----------------------
1590 REM EINGABE DES FILENAMENS
1600 REM ----------------------
```

```
1610 PRINT:PRINT
1620 INPUT"NAME DES FILES:";F$
1630 IFLEN(F$)<=16THEN1650
1640 PRINT"FILENAME ZU LANG!":GOTO1620
1650 GF=0:FL=0
1660 RETURN
```

Dieses Dienstprogramm wurde auf einem CBM 64 erstellt. Es ist jedoch in dieser Version auch auf dem VC 20 lauffähig. Dazu muß lediglich die Zeile 100, die beim CBM 64 die Bildschirmfarben setzt, entsprechend angepasst oder ignoriert werden. Wenn Sie Wert auf eine optisch einwandfreie Bildschirmausgabe legen, können Sie die Zeilen 110-230 der VC 20-Bildschirmdarstellung anpassen.

4.1.3 Backup-Programm - Kopieren von Disketten

Die Floppy VC 1541 hat als Einzellaufwerk nicht die Möglichkeit, selbstständig Disketten zu duplizieren, wie dies die Doppellaufwerke mit dem Befehl 'Duplicate' bzw. 'BACKUP' in BASIC 4.0 bieten. Bei der 1541 muß dies per Programm über den Rechner gemacht werden.

Das Prinzip sieht so aus:
Zuerst werden die BAM sowie Namen und ID der zu kopierenden Diskette gelesen. Aus der BAM ermittelt man nun, welche Blocks auf der Originaldiskette belegt sind. Aus Gründen der Zeitersparnis sollen nur die belegten Blocks kopiert werden. Dann wird eine Direktzugriffsdatei eröffnet von der von den ersten 169 Sektoren (soviel wie in etwa in den Speicher des Commodore 64 passen) die belegten gelesen werden. Dann wird der Benutzer aufgefordert, eine neue Diskette ins Laufwerk zu legen. Diese wird nun mit dem Namen und der ID der Originaldiskette formatiert. Jetzt werden die zuvor gelesenen Blocks aus dem Speicher auf die neue Diskette geschrieben. Nun können die nächsten 169 Blocks der Originaldiskette geprüft und bei Bedarf in den Speicher gelesen und anschließend auf die Zieldiskette geschrieben werden. Dies läuft insgesamt viermal ab, bis die komplette Diskette kopiert ist.

Das Programm ist bis auf das Lesen und Schreiben der Direktzugriffsdatei in BASIC geschrieben. Die dafür enthaltenen Maschinenprogramme sind bedeutend schneller als eine GET#-Schleife über 256 Bytes in BASIC. Da die Effektivität des Programms, die Anzahl der Diskettenwechsel, vom zur Verfügung stehenden freien Speicher des Rechners abhängt, ist es nur für den Commodore 64 gedacht. Selbst bei einem VC 20 mit 16 K Erweiterung wären je elfmaliger Wechsel von Original- und Zieldiskette erforderlich.

Hier noch ein ungefährer Zeitvergleich zwischen diesem Programm und dem Duplizieren auf einem Doppellaufwerk mit der gleichen Kapazität. Unser Programm braucht je nach Diskettenbelegung ca. 20 Minuten, die CBM 4040 schaffts in ca. 3 Minuten.

Das Duplizieren von Disketten mit diesem Programm verläuft denkbar einfach: Sie brauchen nach dem Starten lediglich nach den Anweisungen auf dem Bildschirm jeweils die Original- oder die Zieldiskette einzulegen, den Rest erledigt das Programm für Sie.

```
100 REM BACKUP-PROGRAMM C64 - VC 1541
110 REM
120 POKE56,23:CLR:GOSUB640
130 OPEN1,8,15
140 DIM B%(35,23),S%(35),Z(7),A$(1)
150 A$(0)="ZIEL":A$(1)="ORIGINAL":R=1
160 AD=23*256:GOSUB590
```

```
170 POKE250,0:POKE251,AD/256
180 GOSUB530:GOSUB290
190 PRINTNS"BLOCKS ZU KOPIEREN":PRINT
200 T=1:S=0
210 FORI=1TO4:TT=T:SS=S:R=1:IFI=1THEN240
220 IFR=0ANDI=1THENGOSUB450:GOTO240
230 GOSUB590
240 POKE251,AD/256:FORJ=1TO169
250 IFB%(T,S)=0THENGOSUB570
260 S=S+1:IFS=S%(T)THENT=T+1:S=0:IFT=36THENJ=169
270 NEXT:IFRTHENR=0:T=TT:S=SS:GOTO220
280 NEXT:GOTO510
290 T=18:S=0:GOSUB570
300 NS=0: FOR T=1TO35 : S=0
310 NS=NS+S%(T)-PEEK(AD+4*T)
320 FORJ=1TO3
330 B=PEEK(AD+4*T+J)
340 FORI=0TO7
350 B%(T,S)=B AND Z(I):S=S+1
360 NEXT I,J
370 FOR S=S%(T)TO23
380 B%(T,S)=-1 : NEXT S,T
390 FOR I=0TO15
400 A=PEEK(AD+144+I)
410 IFA<>160THENN$=N$+CHR$(A)
420 NEXT
430 I$=CHR$(PEEK(AD+162))+CHR$(PEEK(AD+163))
440 PRINTN$,I$:RETURN
450 PRINT"BITTE NEUE DISKETTE EINLEGEN"
460 PRINT"UND RETURN DRUECKEN !":PRINT:POKE198,0:CLOSE2
470 GETA$:IFA$<>CHR$(13)THEN470
480 PRINT#1,"NO:"N$","I$
490 INPUT#1,A,B$,C,D:IFATHENPRINTA","B$","C","D:END
500 GOTO630
510 CLOSE2:CLOSE1:END
520 REM SEKTOREN PRO TRACK
530 FORT=1TO35
540 S%(T)=21:IFT>17THENS%(T)=19:IFT>24THENS%(T)=18:
    IFT>30THENS%(T)=17
550 NEXT
560 FORI=0TO7:Z(I)=2↑I:NEXT:RETURN
570 IFRTHENPRINT#1,"U1 2 0"T;S:SYSIN:RETURN
580 PRINT#1,"B-P 2 0":SYSOUT:PRINT#1,"U2 2 0"T;S:RETURN
590 CLOSE2:PRINT"BITTE "A$(R)"DISKETTE EINLEGEN"
600 PRINT"UND RETURN DRUECKEN !":PRINT:POKE198,0
610 GETA$:IFA$<>CHR$(13)THEN610
620 PRINT#1,"I0
630 OPEN2,8,2,"#":RETURN
640 FOR I = 828 TO 873 : REM MASCHINENPROGRAMM LESEN
650 READ X : POKE I,X : S=S+X : NEXT
660 DATA 162,  2, 32,198,255,160,  0, 32,207,255,145,250
670 DATA 200,208,248,230,251, 32,204,255, 96,198,  1,162
680 DATA   2, 32,201,255,160,  0,177,250, 32,210,255,200
690 DATA 208,248,230,251, 32,204,255,230,  1, 96
700 IF S <> 7312 THEN PRINT "FEHLER IN DATAS !!" : END
710 IN=828:OUT=849:RETURN
```

4.1.4 Kopieren einzelner Files auf eine andere Diskette

Das nachfolgende Programm erlaubt es Ihnen, einzelne Dateien von einer Diskette auf eine andere Diskette zu kopieren. Bei den Dateien kann es sich um Programme (PRG), sequentielle Dateien (SEQ) oder Userdateien (USR) handeln. Relative Dateien lassen sich mit diesem Programm nicht kopieren; können jedoch mit einem BASIC-Programm, daß alle Datensätze in ein Stringarray liest und von dort wieder in eine neue Datei schreibt, kopiert werden.

Das Programm liest im ersten Gang die komplette Datei in den Speicher des Commodore 64. Dann wird die Zieldiskette eingelegt und dort eine Datei mit gleichem Namen eröffnet. Dann werden die kompletten Daten auf die zweite Diskette geschrieben. Zum Datenspeichern stehen im Rechner 49 KByte zur Verfügung; Sie können deshalb Dateien mit bis zu 196 Blocks auf Diskette verarbeiten.

Aus Geschwindigkeitsgründen wurde das Einlesen und Zurückschreiben der Daten mit einem kleinen Maschinenprogramm erledigt, das in DATA-Statements abgelegt ist.

Das Programm eignet sich außer zum Kopieren von sequentiellen Dateien wie gesagt auch zum Kopieren von Programmen aller Art; die Startadresse (bei Maschinenprogrammen) ist dabei nicht relevant.

Bei der Bedienung des Programms brauchen Sie sich nur an die Anweisungen zu halten und die entsprechenden Disketten einlegen.

```
100 REM FILE-KOPIERPROGRAMM  C64
110 REM
120 POKE 56,12 : CLR
130 GOSUB 1000
140 INPUT "DATEINAME ";N$
150 PRINT "DATEITYP  ";
160 GETT$:IFT$<>"S"ANDT$<>"P"ANDT$<>"U"THEN160
170 PRINTT$:PRINT
180 PRINT"BITTE ORIGINALDISKETTE EINLEGEN"
190 PRINT"UND TASTE DRUECKEN !":PRINT
200 GETA$:IFA$=""THEN200
210 OPEN 2,8,2,N$+","+T$
220 POKE3,0:POKE4,12:SYS866
230 CLOSE2
240 PRINT"BITTE ZIELDISKETTE EINLEGEN "
250 PRINT"UND TASTE DRUECKEN !":PRINT
260 GETA$:IFA$=""THEN260
270 OPEN 2,8,2,N$+","+T$+",W"
280 POKE3,0:POKE4,12:SYS828
290 CLOSE 2 :END
```

```
1000 FOR I = 828 TO 898
1010 READ X : POKE I,X : S=S+X : NEXT
1020 DATA 162,  2, 32,201,255,198,  1,160,  0, 56,165,  3
1030 DATA 229,  5,165,  4,229,  6,176, 13,177,  3, 32,210
1040 DATA 255,230,  3,208,236,230,  4,208,232,230,  1, 76
1050 DATA 204,255,162,  2, 32,198,255,160,  0, 32,207,255
1060 DATA 145,  3,230,  3,208,  2,230,  4, 36,144, 80,241
1070 DATA 165,  3,133,  5,165,  4,133,  6, 76,204,255
1080 IF S <> 8634 THEN PRINT "FEHLER IN DATAS !!" : END
1090 RETURN
```

4.1.5 Einlesen des Directorys innerhalb von Programmen

Es gibt Anwendungsprogramme, die benutzereigene Dateien unter einem beliebigen Namen abspeichern. Wenn Sie zum Arbeiten mit dieser Datei deren Namen angeben müssen, der Name Ihnen aber entfallen ist, so ergibt sich ein Problem: Zum Auffinden dieses Namens müssen Sie das Programm verlassen, den Namen im Directory suchen und das Programm neu laden und starten. Diese Prozedur läßt sich durch Integrierung einer Directory-Auflistroutine in das Programm vermeiden. Ist Ihnen dann ein Dateiname entfallen, so können Sie, z.B. mit einer Funktionstaste, das Directory auf dem Bildschirm ausgeben, ohne daß das Programm verlassen werden muß. Wir haben eine dementsprechende Routine entwickelt, deren Listing nun folgt:

```
100 PRINTCHR$(147);
110 OPEN15,8,15,"I0":OPEN2,8,2,"#"
120 T=18:S=1
130 PRINT#15,"U1";2;0;T;S
140 PRINT#15,"B-P";2;0
150 GET#2,X$:IFX$=""THENX$=CHR$(0)
160 T=ASC(X$)
170 GET#2,X$:IFX$=""THENX$=CHR$(0)
180 S=ASC(X$)
190 FORX=0TO7
192 PRINT#15,"B-P";2;X*32+2
194 GET#2,X$:IF X$=""THEN X$=CHR$(0)
196 IF ASC(X$)=0 THEN 290
200 PRINT#15,"B-P";2;X*32+5
210 FF$=""
220 FORY=0TO15
230 GET#2,X$:IFX$=""THENX$=CHR$(0)
240 IFASC(X$)=160THEN 270
250 FF$=FF$+X$
260 NEXTY
270 IFA=0THENA=1:PRINTFF$;:GOTO290
280 A=0:PRINTTAB(20);FF$
290 NEXTX
300 IFT<>0THEN130
310 CLOSE1:CLOSE15
320 PRINT"WEITER MIT RETURN!"
330 INPUT X$
340 END:REM WENN UNTERPROGRAMM, DANN RETURN
```

Nach Selektieren der Filenamen aus dem Directory werden diese auf dem Bildschirm ausgegeben. Soll dieses Programm als Unterprogramm benutzt werden, daß mit GOSUB aufgerufen wird, so muß in Zeile 340 anstatt des Befehls END der Befehl RETURN eingesetzt werden.
Diese Routine haben wir auch in den Dienstprogrammen in den Kapiteln 4.1.1 und 4.1.2 verwendet.

4.1.6 Suchen von "versteckten" Files

Viele unsere Leser kennen die Situation: Irgendein Programm oder sogar die gesamte Diskette ist durch eine Fehlbedienung "zerschossen" worden. Das kann unter Umständen schlimme Folgen haben. Die Arbeit von Tagen oder Wochen kann durch einen falschen Befehl umsonst gewesen sein.

Welche Fehlbedienungen können auftreten? Sehr oft passiert es, daß Diskette ohne Angabe der ID formatiert werden. Dies hat zur Folge, daß alle Files auf dieser Diskette gelöscht werden. Löschen ist eigentlich der falsche Ausdruck. Es wird lediglich der Zugriff unterbunden, in dem der erste Block des Directorys (Spur 18, Sektor 1) gelöscht wird. Die ursprünglich vom File belegten Blöcke sind aber noch vorhanden.

Ein anderes Beispiel: Sie haben ein File mit dem Klammeraffen überschrieben. Hier kann es vorkommen, daß das Betriebssystem der Floppy die entsprechende Routine nicht ordnungsgemäß durchführt. Der Fehler, der dann meist auftritt macht sich dadurch bemerkbar, daß beim Laden des Programms "A" das Programm "B" erscheint. Der Grund ist folgender: Anstatt dem Anfangsblock des neu abgespeicherten Programms wurde der Anfangsblock eines anderen Programms in das Directory eingetragen. Das "verlorene" Programm befindet sich noch auf der Diskette, nur der erste Datenblock ist unbekannt.

Das folgende Programm ermittelt alle Datenblöcke der Diskette, die erste Datenblöcke eines Files sein könnten. Ist also ein Programm verloren gegangen, so können Sie durch Vergleich dieser Blöcke mit den tatsächlichen Anfangsblöcken den Anfangsblock des verlorenen Programms ermitteln. Dieser Anfangsblock muß dann mit dem Disk-Monitor in den Directoryeintrag eingesetzt werden. Achtung! Das folgende Programm gibt die Blöcke dezimal aus, der Disk-Monitor jedoch arbeitet hexadezimal.

Bei einer vollkommen gelöschten Diskette müssen Sie nicht nur den ersten Datenblock, sondern den gesamten Fileeintrag erneuern. Da die Anzahl der vom Programm belegten Blöcke nicht bekannt ist, erscheint später im Directory eine 0. Ratsam ist, wenn Sie das rekonstruierte Programm einmal laden und auf einer anderen Diskette abspeichern.

```
5 DIMBL(35,20)
6 PRINT"BITTE DISKETTE EINLEGEN UND RETURN DRÜCKEN"
7 INPUT X$
10 OPEN1,8,15,"I0"
20 OPEN2,8,2,"#"
30 REM ***************
40 REM LESEN SEKTOREN
50 REM ***************
52 : FOR I=1TO4
53 :   READ T1,T2,S1
```

```
54 :    DATA 1,17,20
55 :    DATA 18,24,18
56 :    DATA 25,30,17
57 :    DATA 31,35,16
60 :    FOR T=T1TOT2
65 :    IFT=18THEN140
70 :       FOR S=0TOS1
80 :         PRINT#1,"U1";2;0;T;S
85 :         PRINT#1,"B-P";2;0
90 :         GET#2,B1$:GET#2,B2$
100 :        IF B1$=""THENB1$=CHR$(0)
110 :        IF B2$=""THENB2$=CHR$(0)
115 :        TR=ASC(B1$):SC=ASC(B2$)
120 :        IFBL(T,S)<>2THENBL(T,S)=3
125 :        IF TR=0 OR TR>35 OR SC>20 THEN BL(T,S)=1:GOTO 130
126 :        BL(TR,SC)=2
130 :       NEXT S
140 :   NEXT T
150 :NEXT I
155 CLOSE1:CLOSE2
160 FOR T=1TO35
170 : FOR S=0TO20
180 :    IF BL(T,S)=3THENPRINT"SP:";T;" - SEK:";S,:FI=FI+1
190 : NEXT S
195 :NEXT T
200 PRINT"FILES:"FI
```

Nach dem Start dieses Programms werden die ersten beiden Bytes von jedem Block der Diskette eingelesen. Diese beiden Bytes enthalten die Adresse des nächsten Blocks des entsprechenden Programms. Dies dauert ca. 2 Minuten. Danach ermittelt das Programm alle Blöcke, die selbst ein Datenblock sein könnten und auf die kein anderer Datenblock zeigt. Nur diese Blöcke könnten erste Datenblöcke sein. Befinden sich jedoch relative Dateien auf dieser Diskette, so können Blöcke ausgegeben werden, die keine ersten Datenblöcke sind.

Eine detailierte Beschreibung der Arbeitsweise dieses Programms wäre kompliziert und nur für "Spezialisten" verständlich. Diese jedoch können die Programmlogik direkt aus dem Listing ersehen.

Da auch wir nicht vollkommen sind, gab es Situationen, in denen uns dieses Programm sehr weitergeholfen hat. Wir hoffen, daß auch Sie mit diesem Programm stunden- und tagelange Arbeit retten können.

4.2 Die Dienstprogramme der Test/Demo-Diskette

Es gibt viele VC-5141 Besitzer, die mit den auf der Test/Demo-Diskette enthaltenen Programmen wenig anzufangen wissen. Der Grund dafür ist, daß diese Programme entweder in englischer Sprache selbstdokumentierend oder aber sogar gänzlich undokumentiert sind. Die folgenden Beschreibungen dieser Programme soll Ihnen weiterhelfen:

4.2.1 DOS 5.1

Das DOS 5.1 vereinfacht die Handhabung des VC-1541 DOS. Es ist auf den Rechnern VC-20 und COMMODORE 64 einsetzbar. Zum Laden des DOS 5.1 mit dem VC-20 geben Sie die Befehle

```
LOAD"VIC-20 WEDGE",8
RUN
```

ein. Dies ist das Ladeprogramm des DOS 5.1 für den VC-20. Wollen Sie das DOS 5.1 auf dem COMMODORE 64 betreiben, so geben Sie die Befehle

```
LOAD"C-64 WEDGE",8
RUN
```

ein. Hiermit wird das DOS 5.1 in den CBM 64 geladen.
Doch was bietet nun dieses DOS 5.1? Sie können die am meisten benötigten Befehle mit Symbolen abkürzen. Wollen Sie z....B. das Directoy auf dem Bildschirm anzeigen, so geben Sie den DOS 5.1-Befehl '@$' oder '>$' ein. Hier wird auch nicht das im Speicher befindliche Programm gelöscht.

Die einzelnen Befehle des DOS 5.1:

Schreibweise	Funktion
@$ oder >$	Das Directory wird angezeigt
@V oder >V	Selbe Funktion wie "VALIDATE"
@C:... oder >C:..	Kopieren von Files (COPY)
↑file oder /file	Laden von Programmen (↑ = mit automatisch. Start)
@ oder >	Fehlerkanal abfragen und anzeigen
@N:... oder >N:...	Formatieren einer Diskette
@I oder >I	Initialisieren der Diskette
@R:... oder >R:...	Umbenennen eines File (RENAME)
@S:... oder >S:...	Löschen eines Files (SCRATCH)

4.2.2 COPY/ALL

Mit dem Programm "COPY/ALL" können Files zwischen zwei Laufwerken verschiedener Adressen ausgetauscht werden. Dazu muß ein Laufwerk z.B. mit dem Programm "DISK ADDR CHANGE" auf eine andere Geräteadresse als 8 umgestellt werden. Nach dem Starten des Programms erscheint die Meldung:

```
disk copy all                jim butterfield

from unit? 8
```

auf dem Bildschirm. Hier geben Sie die Geräteadresse der Diskettenstation an, von dem Sie die Files herunterholen möchten. Ist dies die Adresse 8, so drücken Sie nur RETURN. Anschließend geben Sie das entsprechende Laufwerk dieser Diskettenstion an (bei Einzellaufwerken immer 0). Auf diese Weise stellen Sie auch die Geräteadresse des Ziellaufwerkes ein. Ist dies geschehen, so fragt das Programm

```
want to new the output disk
?n
```

Es wird gefragt, ob die Zieldiskette noch formatiert werden soll. Sie antworten hier mit 'y' (ja) oder 'n' (nein).
Dann können Sie die zu kopierende Files mit dem Joker (*) auswählen. Sollen alle Files kopiert werden, so geben Sie nur den Stern ein.
Nun gibt das Programm die Anweisung

```
hold down 'y' or 'n' key to select
```

Das Programm zeigt nun die Files der Originaldiskette an, die Sie dann mit der Taste 'y' (ja) oder 'n' (nein) auswählen können. Die Files, bei der Sie 'y' gedrückt haben, werden kopiert.
Erscheinen während dem Kopiervorgang Sterne (***) hinter den Files, so bedeutet das, daß dieser Kopiervorgang nicht fehlerfrei verlief.
Können nicht alle Files auf die Zieldiskette untergebracht werden, so wird "*** output disk full" und "do you have a new one" gemeldet. Die restlichen Files können auf eine andere, formatierte Diskette untergrbracht werden, in dem Sie nach der Frage 'y' eingeben.
Nach Abschluß des Kopiervorgangs wird die Anzahl der freien Blocks der Zieldiskette angezeigt.

4.1.3 DISK ADRR CHANGE

Mit diesem Programm können Laufwerke softwaremäßig auf eine andere Geräteadresse eingestellt werden (4-15). Nach Starten des Programms schalten Sie alle angeschlossenen Laufwerke, außer dem zu ändernden Laufwerk aus. Nun geben Sie die alte

und anschließend die neu Geräteadresse ein.
Danach wird die Adresse umgestellt und alle anderen Laufwerke können wieder eingeschaltet werden.
Folgende Laufwerke können von diesem Programm umgestellt werden:

1541	DOS V2.6
2031	DOS V2.6
2040	DOS V1.1
4040	DOS V2.1
4040	DOS V2.7
8050	DOS V2.5
8050	DOS V2.7
8250	DOS V2.7

4.2.4 DIR

Dies ist ein kleines Hilfsprogramm mit folgenden Möglichkeiten:

d - Zeigt das Directory auf dem Bildschirm an

> - Mit diesem Zeichen kann ein Diskettenbefehl in verkürzter Form eingegeben werden (z.B >N:TEST,KN) zum Formatieren einer Diskette

q - Verlassen des Programms

s - Fehlerkanal anzeigen

Diese Möglichkeiten haben Sie auch mit dem DOS 5.1, das außerdem noch weitere Befehle beinhaltet.

4.2.5 VIEW BAM

Mit diesem Dienstprogramm können Sie die Belegung der Diskettenblöcke auf dem Bildschirm anzeigen lassen. Diese Tabelle zeigt in vertikaler Richtung die Sektoren und in horizontaler Richtung die Spuren an. Normale Kreuze kennzeichnen freie und reverse Kreuze die belegten Blöcke. Die Bezeichnung "n/a" bedeutet, daß diese Blöcke nicht auf der Spur existieren.
Nach Ausgabe der Tabelle wird der Diskettenname und die Anzahl der freien Blöcke angezeigt.

4.2.6 CHECK DISK

Das Dienstprogramm "CHECK DISK" testet jeden Block der Diskette, indem er beschrieben und gelesen wird. Der momentan bearbeitete Block und die Gesamtzahl der getesteten Blöcke

wird am Bildschirm angezeigt.

4.2.7 DISPLAY T&S

Wenn Sie an dem Aufbau der einzelnen Blocks der Diskette interessiert sind und diese auf dem Bildschirm oder Drucker ausgeben wollen, hilft Ihnen dieses Dienstprogramm weiter. Nach Starten des Programms geben Sie die gewünschte Spur (TRACK) und den Block (SECTOR) ein. Dieser wird dann entweder auf dem Drucker oder auf dem Bildschirm ausgegeben. Der in diesem Buch enthaltene DISK-MONITOR ist aber wesentlich komfortabler als dieses Programm, da mit ihm auch Blöcke geändert und wieder zurückgeschrieben werden können.

4.2.8 PERFORMANCE TEST

Dieses Programm ermöglicht es, die Mechanik des Laufwerkes VC-1541 zu testen. Dazu werden alle Zugriffsbefehle auf die Diskette in folgender Reihenfolge ausgeführt:

1. Diskette wird formatiert
2. Ein File wird zum Schreiben geöffnet
3. Daten werden in dieses File geschrieben
4. Das File wird wieder geschlosssen
5. Dieses File wird zum Lesen geöffnet
6. Die Daten werden gelesen
7. Das File wird wieder geschlossen
8. Das File wird gelöscht
9. Die Spur 35 wird beschrieben
10. Die Spur 1 wird beschrieben
11. Die Spur 35 wird gelesen
12. Die Spur 1 wird gelesen

Nach jedem Zugriff auf die Diskette wird der Fehlerkanal angezeigt. Auf diese Weise kann festgestellt werden, welcher Zugriff auf die Diskette nicht fehlerfrei verläuft.
Benutzen Sie für diesen Test nur Disketten, die keine wichtigen Daten enthalten, da diese sonst verloren gehen.

4.3 BASIC-Erweiterungen und Programme zur komfortablen Nutzung der VC 1541

4.3.1 Eingabe beliebig langer Strings von Diskette

Das Einlesen von Daten von der Floppy mit Hilfe des INPUT#-Befehls hat leider einen großen Nachteil: Mit Commodore 64 und VC 20 können keine Daten eingelesen werden, die mehr als 88 Zeichen haben. Dies liegt am Eingabepuffer des Rechners, der nicht länger ist. Außerdem können nicht alle Zeichen mit INPUT# gelesen werden. Steht innerhalb eines Datensatzes ein Komma oder ein Doppelpunkt, so sieht der Rechners dies als Trennzeichen an, und der Rest der Eingabe wird der nächsten Variable zugewiesen. Enhält der INPUT#-Befehl nur eine Variable, so wird der Rest ganz ignoriert und beim nächsten INPUT# wird erst hinter dem nächsten Carriage Return (CHR$(13)) weitergelesen. Die Alternative, die Eingabe mit dem GET#-Befehl erfordert eine langsame Schleife in BASIC, die wir vermeiden wollen.

Hier kann eine kleine Maschinenroutine Abhilfe schaffen.

Wir ändern hier den INPUT#-Befehl ab, indem wir als zusätzlichen Parameter die Anzahl der zu lesenden Zeichen mit angeben. Zur Unterscheidung vom normalen INPUT#-Befehl nennen wir unseren Befehl INPUT*. Die Syntax sieht dann folgendermaßen aus:

```
INPUT* lf, len, var
```

Dabei ist lf die logische Filenummer der zuvor geöffneten Datei, len ist die Anzahl der Zeichen , die eingelesen werden sollen und var ist die Stringvariable, in die die Zeichen eingelesen werden sollen. Ein Programmausschnitt könnte dann z.B. so aussehen:

```
100 OPEN 2,8,2, "DATEI"
110 INPUT* 2,100,A$
```

Damit wird ein String von 100 Zeichen Länge aus der geöffneten Datei nach A$ gelesen. Dieses Verfahren ist besonders für relative Dateien geeignet, da hierbei nach der Positionierung der Record-Zeigers mit einem Befehl der komplette Datensatz gelesen werden kann. Die Aufteilung des Datensatzes in die einzelnen Datenfelder kann dann mit dem MID$-Befehl geschehen. Wie man Datensätze auf elegante Weise erzeugt, wird im nächsten Kapitel beschrieben.

Bei diesem Verfahren ist es auch nicht mehr nötig, einen Datensatz mit einem Carriage Return abzuschließen. Sie können also besonders bei relativen Dateien die maximale Datensatzlänge ausnutzen:

```
100 OPEN 1,8,15
110 OPEN 2,8,2, "REL-DATEI,L,"+CHR$(20)
```

```
120 PRINT#1, "P"+CHR$(10)+CHR$(0)+CHR$(1)
130 PRINT#2, "12345678901234567890";
140 PRINT#1, "P"+CHR$(10)+CHR$(0)+CHR$(1)
150 INPUT* 2,20,A$
160 PRINT A$

12345678901234567890
```

Anschießend finden Sie das Assemblerlisting des Maschinenprogramms, das im Kassettenpuffer abgelegt wurde sowie je ein Ladeprogramm in BASIC für Commodore 64 und VC 20.

```
110:   033C                .OPT P1
                        ; INPUT* LF,LEN,A$
                        ;
150:   033C          INPUT    =   $85
160:   033C          STERN    =   $AC
170:   033C          BASVEC   =   $308
180:   033C          CHRGET   =   $73
190:   033C          CHRGOT   =   CHRGET + 6
                        ;
210:   033C          ; C64 - VERSION
210:   033C          ;
380:   033C          CHKIN    =   $E11E
390:   033C          BASIN    =   $E112
400:   033C          CHKCOM   =   $AEFD
410:   033C          INTER    =   $A7AE
420:   033C          EXECOLD  =   $A7E7
430:   033C          INPUTOLD =   $ABBF
440:   033C          FINDVAR  =   $B08B
450:   033C          STRRES   =   $B475
460:   033C          FRESTR   =   $B6A3
470:   033C          GETBYT   =   $B79E
                        ;
                        ; 20ER VERSION
                        ;
240:   033C          CHKIN    =   $E11B
250:   033C          BASIN    =   $E10F
260:   033C          CHKCOM   =   $CEFD
270:   033C          INTER    =   $C7AE
280:   033C          EXECOLD  =   $C7E7
290:   033C          INPUTOLD =   $CBBF
300:   033C          FINDVAR  =   $D08B
310:   033C          STRRES   =   $D475
320:   033C          FRESTR   =   $D6A3
330:   033C          GETBYT   =   $D79E
                        ;
                        ; GEMEINSAME LABELS
                        ;
490:   033C          VARADR   =   $49
500:   033C          CLRCH    =   $FFCC
510:   033C          PARA     =   $61
                        ;
530:   033C                   *=   828
540:   033C A9 47    INIT     LDA  #<TEST
```

```
550:   033E A0 03            LDY  #>TEST
560:   0340 8D 08 03         STA  BASVEC
570:   0343 8C 09 03         STY  BASVEC+1
580:   0346 60               RTS
                      ;
600:   0347 20 73 00 TEST    JSR  CHRGET
610:   034A C9 85            CMP  #INPUT
620:   034C F0 06            BEQ  FOUND
630:   034E 20 79 00         JSR  CHRGOT
640:   0351 4C E7 A7         JMP  EXECOLD  ; ZUR ALTEN ROUTINE
650:   0354 20 73 00 FOUND   JSR  CHRGET
660:   0357 C9 AC            CMP  #STERN  ; NEUE INPUT ROUTINE
670:   0359 F0 06            BEQ  OKSTERN
680:   035B 20 BF AB         JSR  INPUTOLD
680:   035E 4C AE A7         JMP  INTER
690:   0361 20 9B B7 OKSTERN JSR  GETBYT-3  ; FILENUMMER HOLEN
700:   0364 20 1E E1         JSR  CHKIN
710:   0367 20 FD AE         JSR  CHKCOM
720:   036A 20 9E B7         JSR  GETBYT  ; LAENGE
730:   036D 8A               TXA
730:   036E 48               PHA          ; MERKEN
740:   036F 20 FD AE         JSR  CHKCOM
750:   0372 20 8B B0         JSR  FINDVAR  ; VARIABLE SUCHEN
760:   0375 85 49            STA  VARADR
760:   0377 84 4A            STY  VARADR+1
770:   0379 20 A3 B6         JSR  FRESTR
780:   037C 68               PLA          ; LAENGE
790:   037D 20 75 B4         JSR  STRRES  ; PLATZ FÜR STRING RESERVIEREN
800:   0380 A0 02            LDY  #2
810:   0382 B9 61 00 STORE   LDA  PARA,Y
820:   0385 91 49            STA  (VARADR),Y
830:   0387 88               DEY
840:   0388 10 F8            BPL  STORE
850:   038A C8               INY          ; Y=0
860:   038B 20 12 E1 FETCH   JSR  BASIN
870:   038E 91 62            STA  (PARA+1),Y
880:   0390 C8               INY
890:   0391 C4 61            CPY  PARA
900:   0393 D0 F6            BNE  FETCH
910:   0395 20 CC FF         JSR  CLRCH
910:   0398 4C AE A7         JMP  INTER   ; ZUR INTERPRETERSCHLEIFE
```

Hier sind nun die BASIC-Programme zur Eingabe der Maschinenprogramme für den INPUT* - Befehl.

INPUT* , 64er Version

```
 100 FOR I = 828 TO 922
 110 READ X : POKE I,X : S=S+X : NEXT
 120 DATA 169, 71,160,  3,141,  8,  3,140,  9,  3, 96, 32
 130 DATA 115,  0,201,133,240,  6, 32,121,  0, 76,231,167
 140 DATA  32,115,  0,201,172,240,  6, 32,191,171, 76,174
 150 DATA 167, 32,155,183, 32, 30,225, 32,253,174, 32,158
```

```
160 DATA 183,138, 72, 32,253,174, 32,139,176,133, 73,132
170 DATA  74, 32,163,182,104, 32,117,180,160,  2,185, 97
180 DATA   0,145, 73,136, 16,248,200, 32, 18,225,145, 98
190 DATA 200,196, 97,208,246, 32,204,255, 76,174,167
200 IF S <> 11096 THEN PRINT "FEHLER IN DATAS !!" : END
210 SYS 828 : PRINT "OK !"
```

INPUT* , 20er Version

```
100 FOR I = 828 TO 922
110 READ X : POKE I,X : S=S+X : NEXT
120 DATA 169, 71,160,  3,141,  8,  3,140,  9,  3, 96, 32
130 DATA 115,  0,201,133,240,  6, 32,121,  0, 76,231,199
140 DATA  32,115,  0,201,172,240,  6, 32,191,203, 76,174
150 DATA 199, 32,155,215, 32, 27,225, 32,253,206, 32,158
160 DATA 215,138, 72, 32,253,206, 32,139,208,133, 73,132
170 DATA  74, 32,163,214,104, 32,117,212,160,  2,185, 97
180 DATA   0,145, 73,136, 16,248,200, 32, 15,225,145, 98
190 DATA 200,196, 97,208,246, 32,204,255, 76,174,199
200 IF S <> 11442 THEN PRINT "FEHLER IN DATAS !!" : END
210 SYS 828 : PRINT "OK !"
```

4.3.2 Komfortables Aufbereiten von Datensätzen

Haben Sie schon einmal mit relativen Dateien gearbeitet, so wissen Sie, daß dort eine feste Datensatzlänge vorgegeben ist. Dieser Datensatz ist meist in mehrere Felder unterteilt, die ebenfalls an festen Positionen innerhalb des Datensatzes beginnen und eine definierte Länge haben.

Geben Sie z.B. in einem Programm einen neuen Datensatz ein, so wird meistens für jedes Feld ein separater INPUT-Befehl verwendet. Bevor nun der komplette Datensatz geschrieben wird, muß er erst richtig zusammengesetzt werden. Jedes Feld muß auf seine Länge geprüft werden. Ist es länger als die vorgesehene Länge des entsprechenden Datenfeldes, muß der Rest abgeschnitten werden. Bei kürzeren Feldern wird man im allgemeinen mit Leerzeichen auf die geforderte Länge auffüllen. Im folgenden werden Ihnen nun zwei neue BASIC-Befehle vorgestellt, die sich für diese Aufgabe hervoragend eignen. Diese neuen Befehle sind in Maschinensprache geschrieben und werden einmal mit einem SYS-Befehl initialisiert. Ab sofort sind sie dann wie alle anderen BASIC-Befehle über Befehlsworte aufzurufen.

Der erste Befehl bekommt den Namen !STR$ und dient zum Erzeugen eines Strings mit der Länge des Datensatzes.

```
A$ = !STR$(100," ")
```

erzeugt einen String mit 100 Leerzeichen und legt ihn in der Variablen A$ ab.

Der nächste Befehl dient nun zum Einsetzen unserer Datenfelder in den oben erzeugten String. Wollen Sie z.B. die Variable N$, die den Nachnamen enthält, als Datenfeld von 25 Zeichen Länge ab Position 1 in den String A$ einsetzen, so sieht unser neuer Befehl so aus:

```
MID$ (A$,1,25) = N$
```

Hier wird der MID$-Befehl als sogenannte Pseudo-Variable auf der linken Seite des Gleichheitszeichens verwendet. Was dabei passiert ist folgendes:

Die Variable N$ ersetzt die ersten 25 Zeichen der Variable A$. Ist die Variable N$ länger als 25 Zeichen, so ist sichergestellt, daß nur 25 Zeichen ersetzt werden und der Rest der Variablen nicht berücksichtigt wird. Ist N$ jedoch kürzer, werden nur so viele Zeichen ersetzt, wie die N$ beinhaltet. Dort bleiben die ursprünglichen Zeichen in A$ (in unserem Falle die Leerzeichen) erhalten. Das ist ganau das, was wir haben wollen. Jetzt können Sie folgendermaßen programmieren:

```
...
200 INPUT "NACHNAME  ";  N$
210 INPUT "VORNAME   ";  V$
220 INPUT "STRASSE   ";  S$
230 INPUT "NUMMER    ";  NR$
240 INPUT "ORT       ";  O$
250 INPUT "PLZ       ";  P$
260 A$ = !STR$ (94, " ")
270 MID$ (A$,1,25)  = N$
280 MID$ (A$,26,20) = V$
290 MID$ (A$,46,20) = S$
300 MID$ (A$,66,5)  = NR$
310 MID$ (A$,71,20) = O$
320 MID$ (A$,91,4)  = P$
330 PRINT#2, A$
...
```

Hier nun das Maschinenprogramm für den Commodore 64.

```
135:   C800                  *=   $C800
140:   C800          CHKAUF  =    $AEFA
150:   C800          CHKZU   =    $AEF7
160:   C800          CHKCOM  =    $AEFD
170:   C800          FRMEVL  =    $AD9E
180:   C800          CHKSTR  =    $AD8F
190:   C800          FRESTR  =    $B6A3
200:   C800          YFAC    =    $B3A2
205:   C800          CHRGET  =    $73
210:   C800          CHRGOT  =    CHRGET+6
220:   C800          GETBYT  =    $B79B
226:   C800          INTEGER =    $B1AA
229:   C800          DESCRPT =    $64
230:   C800          STRADR  =    $62
231:   C800          ADR2    =    $FB
232:   C800          ADR1    =    $FB+2
233:   C800          LEN1    =    3
234:   C800          LEN2    =    4
235:   C800          ANZAHL  =    5
236:   C800          START   =    6
237:   C800          TYPFLAG =    13
238:   C800          STRCODE =    $C4
240:   C800          ILLQUAN =    $B248
241:   C800          SYNTAX  =    $AF08
242:   C800          POSCODE =    $B9
243:   C800          VECTOR  =    $30A
245:   C800          TEMP    =    LEN1
248:   C800 A9 0D             LDA  #<TESTIN
248:   C802 A0 C8             LDY  #>TESTIN
248:   C804 8D 0A 03          STA  VECTOR
248:   C807 8C 0B 03          STY  VECTOR+1
248:   C80A 4C 6B C8          JMP  MIDSTR
250:   C80D A9 00    TESTIN   LDA  #0
250:   C80F 85 0D             STA  TYPFLAG
250:   C811 20 73 00          JSR  CHRGET
251:   C814 C9 21             CMP  #"!"
251:   C816 F0 06             BEQ  TEST2
```

```
251:  C818 20 79 00          JSR  CHRGOT
251:  C81B 4C 8D AE          JMP  $AE8D
252:  C81E 20 73 00 TEST2    JSR  CHRGET
252:  C821 C9 C4             CMP  #STRCODE
252:  C823 F0 03             BEQ  STRING
253:  C825 4C 08 AF          JMP  SYNTAX
                        ;
                        ; STRING$-FUNKTION
                        ;
900:  C828 20 73 00 STRING   JSR  CHRGET
900:  C82B 20 FA AE          JSR  CHKAUF  ; KLAMMER AUF
910:  C82E 20 9E B7          JSR  GETBYT+3
920:  C831 8A                TXA
920:  C832 48                PHA          ; LÄNGE MERKEN
930:  C833 20 FD AE          JSR  CHKCOM
940:  C836 20 9E AD          JSR  FRMEVL
950:  C839 24 0D             BIT  TYPFLAG
960:  C83B 30 0C             BMI  STR     ; STRING
970:  C83D 20 AA B1          JSR  INTEGER
980:  C840 A5 64             LDA  DESCRPT  ; HIGHBYTE
990:  C842 D0 24             BNE  ILL     ; > 255
1000: C844 A5 65             LDA  DESCRPT+1  ; LOW-BYTE, LÄNGE
1010: C846 4C 52 C8          JMP  STR2
1020: C849 20 82 B7 STR      JSR  $B782   ; SETSTR, TYPFLAG AUF NUMERISCH
1030: C84C F0 1A             BEQ  ILL     ; LÄNGE NULL
1040: C84E A0 00             LDY  #0
1050: C850 B1 22             LDA  ($22),Y  ; ERSTES ZEICHEN
1060: C852 85 03    STR2     STA  TEMP
1070: C854 68                PLA          ; LÄNGE
1080: C855 20 7D B4          JSR  $B47D   ; FRESTR
1090: C858 A8                TAY
1100: C859 F0 07             BEQ  STR3
1110: C85B A5 03             LDA  TEMP
1120: C85D 88       LOOP     DEY
1120: C85E 91 62             STA  (STRADR),Y  ; STRING ERZEUGEN
1130: C860 D0 FB             BNE  LOOP
1140: C862 20 CA B4 STR3     JSR  $B4CA   ; STRING IN DESCRIPTORSTACK BRINGEN
1150: C865 4C F7 AE          JMP  CHKZU
1160: C868 4C 48 B2 ILL      JMP  ILLQUAN
                        ;
                        ; MID$(STRINGVARIABLE,POSITION,LÄNGE) = STRINGAUSDRUCK
                        ; MID$(STRINGVARIABLE,POSITION) = STRINGAUSDRUCK
                        ;
200:  C86B          MIDCODE  =   $CA
210:  C86B          EXECUT   =   $308    ; VECTOR FüR STATEMENT AUSFÜHREN
240:  C86B          EXECOLD  =   $A7E7
250:  C86B          VARNAM   =   $45
255:  C86B          VARADR   =   $49
260:  C86B          DESCRPT  =   $64
270:  C86B          TESTSTR  =   $AD8F
280:  C86B          GETVAR   =   $B08B
290:  C86B          SETSTR   =   $AA52
325:  C86B          TEST     =   $AEFF
330:  C86B          GETBYT   =   $B79E
355:  0003                   *=   3
360:  0004          LAENGE   *=   *+1
```

```
370:  0005           POSITION *=   *+1
372:  0007           VARSTR   *=   *+2
375:  0007           GLEICH   =    $B2
378:  0007           ZEIG2    =    $50
                      ;
400:  C86B A9 76     MIDSTR   LDA  #<MIDTEST
410:  C86D A0 C8              LDY  #>MIDTEST
420:  C86F 8D 08 03           STA  EXECUT
430:  C872 8C 09 03           STY  EXECUT+1
440:  C875 60                 RTS
450:  C876 20 73 00 MIDTEST   JSR  CHRGET
460:  C879 C9 CA              CMP  #MIDCODE  ; CODE FÜR MID$ ?
470:  C87B F0 06              BEQ  MID       ; JA
480:  C87D 20 79 00           JSR  CHRGOT
490:  C880 4C E7 A7           JMP  EXECOLD   ; NORMALES STATEMENT AUSFÜHREN
500:  C883 20 73 00 MID       JSR  CHRGET    ; NÄCHSTES ZEICHEN
505:  C886 20 FA AE           JSR  CHKAUF    ; KLAMMER AUF
510:  C889 20 8B B0           JSR  GETVAR    ; VARIABLE HOLEN
520:  C88C 85 64              STA  DESCRPT
530:  C88E 84 65              STY  DESCRPT+1
535:  C890 85 49              STA  VARADR
535:  C892 84 4A              STY  VARADR+1
540:  C894 20 A3 B6           JSR  FRESTR
545:  C897 A0 00              LDY  #0
545:  C899 B1 64              LDA  (DESCRPT),Y
545:  C89B 48                 PHA            ; LÄNGE
545:  C89C F0 2E              BEQ  ILL
550:  C89E 20 52 AA           JSR  SETSTR    ; STRING IN RAM ÜBERTRAGEN
560:  C8A1 A0 01              LDY  #1
560:  C8A3 B1 49              LDA  (VARADR),Y
560:  C8A5 85 05              STA  VARSTR    ; VARIABLENADRESSE MERKEN
570:  C8A7 C8                 INY
570:  C8A8 B1 49              LDA  (VARADR),Y
570:  C8AA 85 06              STA  VARSTR+1
600:  C8AC 20 FD AE           JSR  CHKCOM
610:  C8AF 20 9E B7           JSR  GETBYT    ; POSITION HOLEN
620:  C8B2 8A                 TXA
630:  C8B3 F0 17              BEQ  ILL
650:  C8B5 CA                 DEX
650:  C8B6 86 04              STX  POSITION
660:  C8B8 20 79 00           JSR  CHRGOT
660:  C8BB C9 29              CMP  #")"      ; AUSDRUCK ZU ENDE ?
665:  C8BD D0 04              BNE  NEXT
665:  C8BF A9 FF              LDA  #$FF      ; MAX. LÄNGE
665:  C8C1 D0 0C              BNE  STORE
670:  C8C3 20 FD AE NEXT      JSR  CHKCOM
670:  C8C6 20 9E B7           JSR  GETBYT    ; LÄNGE HOLEN
680:  C8C9 8A                 TXA
690:  C8CA D0 03              BNE  *+5
700:  C8CC 4C 48 B2 ILL       JMP  ILLQUAN
710:  C8CF 85 03    STORE     STA  LAENGE
715:  C8D1 68                 PLA
715:  C8D2 38                 SEC
715:  C8D3 E5 04              SBC  POSITION
717:  C8D5 C5 03              CMP  LAENGE
717:  C8D7 B0 02              BCS  OK
```

```
717:   C8D9 85 03            STA  LAENGE
720:   C8DB 20 F7 AE OK      JSR  CHKZU   ; KLAMMER ZU
730:   C8DE A9 B2            LDA  #GLEICH
770:   C8E0 20 FF AE         JSR  TEST
780:   C8E3 20 9E AD         JSR  FRMEVL  ; AUSDRUCK HOLEN
790:   C8E6 20 A3 B6         JSR  FRESTR
800:   C8E9 A0 02            LDY  #2
800:   C8EB B1 64            LDA  (DESCRPT),Y
800:   C8ED 85 51            STA  ZEIG2+1
800:   C8EF 88               DEY
800:   C8F0 B1 64            LDA  (DESCRPT),Y
800:   C8F2 85 50            STA  ZEIG2
810:   C8F4 88               DEY
810:   C8F5 B1 64            LDA  (DESCRPT),Y
820:   C8F7 F0 D3            BEQ  ILL     ; NULL DANN FEHLER
840:   C8F9 C5 03            CMP  LAENGE
850:   C8FB B0 02            BCS  OK1
860:   C8FD 85 03            STA  LAENGE
870:   C8FF A5 05    OK1     LDA  VARSTR
880:   C901 18               CLC
880:   C902 65 04            ADC  POSITION
910:   C904 85 05            STA  VARSTR
910:   C906 90 02            BCC  *+4
920:   C908 E6 06            INC  VARSTR+1
940:   C90A A4 03            LDY  LAENGE
950:   C90C 88       LOOP    DEY
950:   C90D B1 50            LDA  (ZEIG2),Y  ; ZEICHEN AUS STRINGAUSDRUCK
960:   C90F 91 05            STA  (VARSTR),Y  ; IN STRINGVARIABLE ÜBERTRAGEN
970:   C911 C0 00            CPY  #0
970:   C913 D0 F7            BNE  LOOP
980:   C915 4C AE A7         JMP  $A7AE   ; ZUR INTERPRETERSCHLEIFE
```

Für diejenigen, die über keinen Monitor oder Assembler für den Commodore 64 verfügen, haben wir ein Ladeprogramm in BASIC abgedruckt.

```
100 FOR I = 51200 TO 51479
110 READ X : POKE I,X : S=S+X : NEXT
120 DATA 169, 13,160,200,141, 10,  3,140, 11,  3, 76,107
130 DATA 200,169,  0,133, 13, 32,115,  0,201, 33,240,  6
140 DATA  32,121,  0, 76,141,174, 32,115,  0,201,196,240
150 DATA   3, 76,  8,175, 32,115,  0, 32,250,174, 32,158
160 DATA 183,138, 72, 32,253,174, 32,158,173, 36, 13, 48
170 DATA  12, 32,170,177,165,100,208, 36,165,101, 76, 82
180 DATA 200, 32,130,183,240, 26,160,  0,177, 34,133,  3
190 DATA 104, 32,125,180,168,240,  7,165,  3,136,145, 98
200 DATA 208,251, 32,202,180, 76,247,174, 76, 72,178,169
210 DATA 118,160,200,141,  8,  3,140,  9,  3, 96, 32,115
220 DATA   0,201,202,240,  6, 32,121,  0, 76,231,167, 32
230 DATA 115,  0, 32,250,174, 32,139,176,133,100,132,101
240 DATA 133, 73,132, 74, 32,163,182,160,  0,177,100, 72
250 DATA 240, 46, 32, 82,170,160,  1,177, 73,133,  5,200
260 DATA 177, 73,133,  6, 32,253,174, 32,158,183,138,240
270 DATA  23,202,134,  4, 32,121,  0,201, 41,208,  4,169
280 DATA 255,208, 12, 32,253,174, 32,158,183,138,208,  3
```

```
290 DATA  76, 72,178,133,  3,104, 56,229,  4,197,  3,176
300 DATA   2,133,  3, 32,247,174,169,178, 32,255,174, 32
310 DATA 158,173, 32,163,182,160,  2,177,100,133, 81,136
320 DATA 177,100,133, 80,136,177,100,240,211,197,  3,176
330 DATA   2,133,  3,165,  5, 24,101,  4,133,  5,144,  2
340 DATA 230,  6,164,  3,136,177, 80,145,  5,192,  0,208
350 DATA 247, 76,174,167
360 IF S <> 31128 THEN PRINT "FEHLER IN DATAS !!" : END
370 SYS 51200 : PRINT "OK !"
```

4.3.3 Spooling - Direktes Drucken von Diskette

Haben Sie an Ihrem Rechner außer der Floppy noch einen Drucker angeschlossen, können Sie eine spezielle Eigenschaft des seriellen Bus ausnutzen.

Es besteht nämlich die Möglichkeit, Dateien von der Floppy direkt zum Drucker zu schicken, ohne daß dies Byte für Byte über den Rechner geschehen muß. Hat man z.B. einen beliebigen Text als sequentielle Datei auf Diskette gespeichert und will man diesen auf dem Drucker ausgeben, wäre folgende Programmierung möglich:

```
100 OPEN 1,4 : REM DRUCKER
110 OPEN 2,8,2, "0:TEXT" : REM TEXTDATEI
120 GET#2, A$ : IF ST = 64 THEN 140
130 PRINT#1, A$; : GOTO 120
140 CLOSE 1 : CLOSE 2
150 END
```

Es werden solange Zeichen von der Floppy geholt und zum Drucker geschickt, bis das Dateiende erkannt wird. Dann werden beide Dateien geschlossen und das Programm beendet.

Beim Spooling wird nun folgendes gemacht:

Zuerst werden wieder beide Dateien geöffnet. Jetzt wird an den Drucker ein Befehl zum Daten empfangen (Listen) gesandt, während die Floppy den Befehl zum Daten senden (Talk) erhält. Ab sofort schickt die Floppy solange Daten selbsttätig an den Drucker, bis das Dateiende errreicht ist. Während dieser Zeit können Sie Ihren Rechner weiter benutzen, ohne daß die Übertragung davon beeinträchtigt wird. Lediglich die Benutzung von Peripheriegeräten über den seriellen Bus ist während dieser Zeit nicht möglich.

In der Praxis wird dies mit einem kleinen Maschinenprogramm gemacht. Will man das Spooling starten, ruft man das Programm auf und übergibt dabei den Namen der Datei, die man senden will.

```
SYS 828, "TEXT"
```

öffnet die Datei "TEXT" auf der Diskette und schickt sie zum Drucker. Sobald die Übertragung startet, meldet sich der Rechner mit 'READY.' wieder und Sie können ihn weiter benutzen, solange nicht auf den seriellen Bus zugegriffen wird. Sie können zum Beweis, daß der Rechner nicht mehr zur Übertragung gebraucht wird, das Buskabel zur Floppy herausziehen, so daß die Floppy nur noch mit dem Drucker verbunden ist. Ist das Spooling beendet, so bleibt die Datei in der Floppy noch geöffnet (die rote LED leuchtet weiter). Sie können die Datei schließen, indem Sie den Drucker kurz aus- und wieder einschalten und den SYS-Befehl ohne Dateinamen eingeben (natürlich muß das Kabel zur Floppy

wieder eingesteckt sein).

```
SYS 828
```

Mit dem gleichen Befehl können Sie auch eine laufende Übertragung beenden. Das Maschinenprogramm sowie je ein Ladeprogramm für Commodore 64 und VC 20 finden Sie im Anschluß.

Hier noch einige Hinweise zur Benutzung:

Wir haben das Druckerspooling mit einem Commodore 64 und einem VC-20 sowie einem Drucker Sheikosha GP 100-VC bzw. VC 1525 erfolgreich durchgeführt. Entsprechende Versuche mit einem EPSON-Drucker mit VC-Interface sowie mit einem VC 1526 fielen negativ aus. Der serielle Bus scheint im Gegensatz zum parallelen IEC-Bus nur mit Einschränkungen spoolfähig zu sein. Deswegen ist auch nach beendetem Spooling ein Ausschalten des Druckers erforderlich, da er sonst den Bus blockiert. Wir würden uns freuen, wenn Sie uns über entsprechende Erfahrungen mit anderen Druckern berichten würden.

```
                    ; 1541 - 64 SPOOL
                    ;
110:   033C         CHRGOT   =   $79
130:   033C         LISTEN   =   $FFB1
140:   033C         ATNRES   =   $EDBE   ; ATN HI
142:   033C         CLOCK    =   $EE85   ; CLOCK HI
144:   033C         DATA     =   $EE97   ; DATA HI
160:   033C         CLOSE    =   $FFC3
170:   033C         CLALL    =   $FFE7
175:   033C         SETFIL   =   $FFBA
180:   033C         GETNAME  =   $E254   ; FILENAMEN HOLEN
190:   033C         OPEN     =   $FFC0
200:   033C         CHKIN    =   $FFC6
202:   033C         UNTALK   =   $FFAB
204:   033C         UNLISTEN =   $FFAE
230:   033C         FNLEN    =   $B7     ;
240:   033C         INDEV    =   $99     ; EINGABEGERAET
260:   033C         NMBFLS   =   $98     ; ANZAHL FILES
280:   033C         ERROR    =   $AF08   ; SYNTAX ERROR
                    ;
300:   033C                      *=   828
310:   033C 20 79 00             JSR  CHRGOT  ; WEITERE ZEICHEN
320:   033F F0 33                BEQ  OFF     ; SPOOLING BEENDEN
330:   0341 20 E7 FF             JSR  CLALL
340:   0344 20 54 E2             JSR  GETNAME
350:   0347 A6 B7                LDX  FNLEN
360:   0349 F0 38                BEQ  SYNTAX
370:   034B A9 02                LDA  #2
380:   034D A2 08                LDX  #8
390:   034F A0 02                LDY  #2
400:   0351 20 BA FF             JSR  SETFIL
410:   0354 20 C0 FF             JSR  OPEN    ; FILE ÖFFNEN
411:   0357 A9 04                LDA  #4
```

```
412:   0359 20 B1 FF          JSR  LISTEN  ; DRUCKER
413:   035C 20 BE ED          JSR  ATNRES
420:   035F A2 02             LDX  #2
430:   0361 20 C6 FF          JSR  CHKIN   ; FLOPPY
435:   0364 20 BE ED          JSR  ATNRES
435:   0367 20 85 EE          JSR  CLOCK
435:   036A 20 97 EE          JSR  DATA
510:   036D A9 00             LDA  #0
520:   036F 85 99             STA  INDEV
530:   0371 85 98             STA  NMBFLS
540:   0373 60                RTS
550:   0374 A9 01    OFF      LDA  #1
560:   0376 85 98             STA  NMBFLS
570:   0378 20 AE FF          JSR  UNLISTEN
580:   037B 20 AB FF          JSR  UNTALK
620:   037E A9 02             LDA  #2
630:   0380 4C C3 FF          JMP  CLOSE
640:   0383 4C 08 AF SYNTAX   JMP  ERROR
```

Hier nun das BASIC-Ladeprogramm für den Commodore 64.

```
100 FOR I = 828 TO 901
110 READ X : POKE I,X : S=S+X : NEXT
120 DATA  32,121,  0,240, 51, 32,231,255,  32, 84,226,166
130 DATA 183,240, 56,169,  2,162,  8,160,  2, 32,186,255
140 DATA  32,192,255,169,  4, 32,177,255, 32,190,237,162
150 DATA   2, 32,198,255, 32,190,237, 32,133,238, 32,151
160 DATA 238,169,  0,133,153,133,152, 96,169,  1,133,152
170 DATA  32,174,255, 32,171,255,169,  2, 76,195,255, 76
180 DATA   8,175
190 IF S <> 9598 THEN PRINT "FEHLER IN DATAS !!" : END
200 PRINT "OK"
```

Für den VC 20 benutzen Sie bitte folgendes Ladeprogramm:

```
100 FOR I = 828 TO 901
110 READ X : POKE I,X : S=S+X : NEXT
120 DATA  32,121,  0,240, 51, 32,231,255,  32, 81,226,166
130 DATA 183,240, 56,169,  2,162,  8,160,  2, 32,186,255
140 DATA  32,192,255,169,  4, 32,177,255, 32,197,238,162
150 DATA   2, 32,198,255, 32,197,238, 32,132,239, 32,160
160 DATA 228,169,  0,133,153,133,152, 96,169,  1,133,152
170 DATA  32,174,255, 32,171,255,169,  2, 76,195,255, 76
180 DATA   8,207
190 IF S <> 9642 THEN PRINT "FEHLER IN DATAS !!" : END
200 PRINT "OK"
```

4.4 Overlaytechnik und Nachladen von Maschinenprogrammen

Eine bewährte Programmiertechnik besteht darin, für eine Problemlösung ein sogenanntes Menue- oder Auswahlprogramm zu schreiben, von dem aus für die einzelnen Teilprobleme jeweils ein eigenes Programm geladen und ausgeführt wird. Dabei gibt es zwei grundsätzliche Möglichkeiten: Übernahme oder keine Übernahme der Variablen in das nachgeladene Programm. Eine Übernahme der Variablen ist nur dann möglich, wenn das aufrufende Programm mindestens so groß oder größer als das nachgeladene ist. Wird von einem Programm aus ein anderes Programm nachgeladen, so bleiben die Zeiger auf das Programmende erhalten und das neue Programm wird vom Beginn an abgearbeitet. In unserem Beispiel würden wir folgendes Ergebnis erhalten:

```
100 REM PROGRAMM 1
110 REM DIESES PROGRAMM IST GRÖSSER ALS DAS ZWEITE
120 A = 1000
130 LOAD "PROGRAMM 2",8

100 REM PROGRAMM 2
110 PRINT A

1000
```

Ist das nachgeladene Programm jedoch größer als das ursprüngliche Programm, so würde ein Teil der Variablen überschrieben und wir erhielten undefinierte Werte. Außerdem würde bei Wertzuweisungen an Variablen der Teil des Programms zerstört, der über die Länge des ersten Programms hinaus geht.

Beim Übernehmen vom Variablen gibt jedoch noch zwei Besonderheiten zu beachten: Handelt es sich um Stringvariablen, die im ersten Programm als Konstanten in Anführungszeichen definiert worden sind, so gibt es Probleme. Bei Stringvariablen wird ein Zeiger verwendet, der auf den eigentlichen Text der Variablen zeigt. Wird eine Stringvaribable nun z.B. mit folgender Anweisung im ersten Programm definiert

```
100 A$ = "TEXT"
```

so zeigt der Variablenzeiger in den Programmtext. Beim Nachladen des nächsten Programms wird nun dieser Zeiger nicht verändert. An der ursprünglichen Stelle steht jetzt jedoch der neue Programmtext, so daß die Variable nun einen undefinierten Inhalt hat. Dies können wir jedoch leicht umgehen. Wir brauchen bloß dafür zu sorgen, daß der Text aus dem Programm in den oberen RAM-Bereich kopiert wird, in dem die Textvariablen normalerweise stehen. Dies erreichen wir z.B. durch folgende Programmzeile:

```
100 A$   "TEXT" + ""
```

Durch die Addition des Leerstrings wird das Kopieren des Variableninhalts in den Stringbereich erzwungen. Ähnliche Überlegungen gelten auch bei Funktionsdefinitionen, da auch hier der Zeiger auf die Definition im Programm zeigt. Hier müssen wir die Funktion im zweiten Programm noch einmal definieren, z.B.

```
100 DEF FN A(X) = 0.5 * EXP (-X*X)
```

Halten wir noch einmal fest:
Wollen wir ein Programm nachladen, so können wir die Variablen nur dann weiter benutzen, wenn das zweite Programm kleiner als das erste Programm ist. Ist das nachgeladene Programm größer und sollen keine Variablen übernommen werden, können wir uns mit einem Trick aus der Affäre ziehen:

Wir brauchen lediglich unmittelbar nach dem Laden den Zeiger auf das Ende des BASIC-Programms auf den Wert des neuen Programms setzen. Dies ist mit zwei POKE-Befehlen möglich, da die Endadresse nach dem Laden zur Verfügung steht:

```
POKE 45, PEEK(174) : POKE 46, PEEK(175) : CLR
```

Der CLR-Befehl ist unbedingt erforderlich. Diese Zeile sollte als erste im nachgeladenen Programm stehen. Damit haben wir also die Möglichkeit geschaffen, beliebig große Programme ohne Variablenübergabe nachzuladen. Eine andere nicht so elegante Möglichkeit besteht darin, den Ladebefehl in den Tastaturpuffer zu schreiben und das Programm dann im Direktmodus automatisch nachladen zu können. Dazu schreiben wir vor dem Laden den LOAD- und RUN-Befehl auf den Bildschirm und füllen den Tastaturpuffer mit 'HOME' und Carriage Return. Im Programm muß danach eine END-Anweisung stehen. Das Betriebssystem holt dann im Direktmodus den Inhalt des Tastaturpuffers und liest damit den LOAD- und RUN-Befehl, der zum Laden und Ausführen des Programms führt. Da dies im Direktmodus geschieht, werden automatisch die Endadresse des Programms gesetzt, die Variablen gelöscht und mit dem nachfolgendem RUN das Programm gestartet. Der Nachteil hierbei ist jedoch, daß der Ladebefehl auf den Bildschirm geschrieben und eine evtl. Bildschirmmaske dabei zerstört wird. In der Praxis sähe das so aus:

```
....
1000 PRINT CHR$(147)"LOAD"CHR$(34)"PROGRAMM 2"CHR$(34)",8"
1010 PRINT : PRINT : PRINT : PRINT
1020 PRINT "RUN"
1030 POKE 631,19 : POKE 632,13 : POKE 633,13
1040 POKE 634,13 : POKE 635,13 : POKE 636,13
1050 POKE 198,6  : END
```

Sie sehen schon, daß dieses Verfahren umständlicher als das oben geschilderte Verfahren ist; es ist nur der Vollständigkeit halber erwähnt. Beim ersten Verfahren wäre nur in Zeile 1000 der programmierte LOAD-Befehl erforderlich:

```
1000 LOAD "PROGRAMM 2",8
```

Beim Nachladen von Maschinenprogrammen hat sich eine andere Technik bewährt.

Werden von einem BASIC-Programm Maschinenprogramme benutzt, so werden diese meist zu Beginn des BASIC-Programms geladen. Dabei müssen wir jedoch zweierlei beachten:

Zum einen muß dafür gesorgt werden, daß die Maschinenprogramme absolut, d.h. in einen bestimmten Speicherbereich geladen werden. Gibt man beim Laden eines Programms keine zusätzlichen Parameter an, so geht das Betriebssytem davon aus, daß es sich um BASIC-Programme handelt und lädt sie immer ab der Startadresse des BASIC-RAMs, im allgemeinen also ab Adresse 2049 (beim Commodore 64). Maschinenprogramme sind jedoch nur lauffähig, wenn sie an die Adresse geladen werden, für die sie auch geschrieben wurden. Dieses absolute Laden kann man durch Anhängen der Sekundäradresse 1 erreichen:

```
LOAD "MASCH-PRG",8,1
```

Desweiteren erinnern wir uns, daß beim Laden im Programmmodus das Programm wieder von Anfang an gestartet wird. Das würde beim Laden von Maschinenprogrammen jedoch zu einer Endlosschleife führen, da das Betriebssystem davon ausgeht, daß ein neues BASIC-Programm nachgeladen wurde:

```
100 LOAD "MASCH-PRG",8,1
```

Hier können wir nun ausnutzen, daß die Variablen beim Nachladen erhalten bleiben. Wenn wir folgendermaßen programmieren, haben wir unser Ziel erreicht:

```
100 IF A=0 THEN A=1 : LOAD "MASCH-PRG",8,1
110 ...
```

Wenn wir das Programm mit RUN starten, hat A noch den Wert Null und die Anweisungen hinter THEN werden ausgeführt: A erhält den Wert 1 und das Maschinenprogramm wird geladen. Wenn nach dem Laden das Programm wieder von Angang an abgearbeitet wird, hat A den Wert 1 und es wird direkt in die nächste Zeile gesprungen, wir haben unser Ziel erreicht.

Ganz ähnlich kann man vorgehen, wenn man mehrere Maschinenprogramme zu laden hat.

```
100 IF A=0 THEN A=1 : LOAD "PROG 1",8,1
110 IF A=1 THEN A=2 : LOAD "PROG 2",8,1
120 IF A=2 THEN A=3 : LOAD "PROG 3",8,1
130 ....
```

Hier wird im ersten Durchlauf PROG 1 geladen, im nächsten PROG 2 usw. Sind alle Programme geladen, wird der Rest des BASIC-Programms ausfgeführt.

4.5 Merge - Aneinanderhängen von BASIC-Programmen

Sicher haben Sie schon einmal an die Möglichkeit gedacht, BASIC-Programme, die Sie einzeln auf Diskette abgespeichert haben, zu einem Programm zusammen zu fügen. Ohne weiteres ist dies nicht möglich, da bei Laden eines Programms das im Speicher stehende Programm überschrieben wird. Mit der Kenntnis, wie BASIC-Programme im Speicher abgelegt und auf Diskette abgespeichert werden, können wir jedoch ein einfaches Verfahren entwickeln, das diese Aufgabe bewerkstelligt.

BASIC-Programme stehen folgendermaßen in Speicher:

```
NL NH   Zeiger auf die nächste Programmzeile, lo hi
ZL ZH   Zeilennummer, lo hi
XX YY ZZ  .....  Zeileninhalt
00      Kennzeichen für Zeilenende

NL NH   Zeiger auf die nächste Programmzeile, lo hi
ZL ZH   Zeilennummer, lo hi
XX YY ZZ  .....  Zeileninhalt
00      Kennzeichen für Zeilenende
```

Am Ende des Programms stehen zusätzlich noch 2 Nullbytes:
00 00 , insgesamt also 3 Nullbytes

In diesem Format werden Programme nun auch abgespeichert. Wo Programmstart und Programmende liegen, steht in zwei Zeigern in der Zeropage:

```
PRINT PEEK(43) + 256 * PEEK(44)
```

ergibt den BASIC-Start, 2049 beim Commodore 64,

```
PRINT PEEK(45) + 256 * PEEK(46)
```

zeigt auf das Byte hinter den drei Nullbytes.

Da ein Programm immer an den BASIC-Start geladen wird, der durch den Zeiger 43/44 bestimmt ist, kann man also erreichen, daß ein zweites Programm ans Ende des ersten Programms geladen wird. In der Praxis müssen wir also wie folgt vorgehen:

Zuerst laden wir das erste Programm in den Speicher. Jetzt holen wir uns den Wert des Programmendes.

```
A = PEEK(45) + 256 * PEEK(46)
```

Dieser Wert wird um 2 vermindert, damit nachher die beiden Nullbytes am Programmende mit überschrieben werden.

```
A = A -2
```

Nun merken wir uns den ursprünglichen Wert des BASIC-Starts.

```
PRINT PEEK(43), PEEK(44)
```

Jetzt setzen wir diesen Wert als BASIC-Start.

```
POKE 43, A AND 255 : POKE 44, A / 256
```

Nun können wir das zweite Programm nachladen.

```
LOAD "PROGRAMM 2",8
```

Wenn wir nun die alten Werte für den BASIC-Start wieder setzen, z.B. 1 und 8 beim Commodore 64 (wie oben mit dem PRINT-Befehl erhalten), haben wir das komplette Programm im Speicher und können es uns mit LIST ansehen oder mit SAVE komplett abspeichern.

```
POKE 43,1 : POKE 44,8
```

Bei dieser Methode ist jedoch folgendes zu beachten:

Das angehangene Programm darf nur Zeilennummern enthalten, die größer sind als die größte Zeilennummer des ersten Programm, da andernfalls diese Zeilennummer nie mit GOTO oder GOSUB erreicht werden könnten und die geordnete Reihenfolge nicht gewährleistet wäre.

Dieses Verfahren eignet sich vor allem zum Anlegen einer Unterprogrammbibliothek für öfter gebrauchte Routinen, die dann nicht jedesmal neu programmiert werden müssen. Gehen Sie beim Anlegen Ihrer Programmbibliothek am besten so vor, daß Sie für jedes Programm einen bestimmten Zeilennummernbereich reservieren, z. B. 20000 - 25000, 25000 - 30000 usw. Wollen Sie mehrere Programme in der oben beschriebenen Weise nachladen, müssen Sie zuerst die Programme mit den kleinsten Zeilennummer laden und danach das Programm mit den nächst höheren Nummern.

4.6 Disk-Monitor für Commodore 64 und VC 20

In diesem Kapitel stellen wir Ihnen ein sehr nützliches Werkzeug für den Umgang mit Ihrer Floppy vor, das Sie in die Lage versetzt, jeden beliebigen Block von Diskette zu laden, auf dem Bildschirm anzuzeigen, zu ändern und wieder auf Diskette zurück zu schreiben.

Das Programm ist aus Geschwindigkeitsgründen vollkommen in Maschinensprache geschrieben. Folgende Befehle werden Ihnen zur Verfügung gestellt:

* Lesen eines Blocks von Diskette
* Schreiben eines Blocks auf Diskette
* Anzeige eines Blocks auf dem Bildschirm
* Ändern eines Blocks auf dem Bildschirm
* Senden von Diskettenbefehlen
* Anzeigen der Fehlermeldung der Diskette
* Rückkehr zu BASIC

Das Programm meldet sich nach dem Starten (automatisch durch das BASIC-Ladeprogramm) mit

```
DISK-MONITOR V1.0
>
```

und erwartet Ihre Eingabe. Geben Sie jetzt '@' (Klammeraffe) ein, so wird die Fehlermeldung von Diskette geholt und auf dem Bildschirm angezeigt, z.B.

```
00, OK,00,00
```

Wollen Sie einen Befehl an die Diskette senden, so geben Sie '@' gefolgt von dem Befehl ein.
Initialisieren können Sie die Diskette dann mit

```
>@I
```

Sie können so sämtliche Diskettenbefehle senden, die Sie sonst über die Befehlsfolge

```
OPEN 15,8,15
PRINT# 15, "Befehl"
CLOSE 15
```

senden würden. Sie können z.B. Dateien löschen, Disketten formatieren usw.

Die wichtigste Funktion des Diskettenmonitors ist jedoch der Direktzugriff auf jeden Block der Diskette. Dazu dienen die Befehle 'R' und 'W'. 'R' steht für READ und liest einen gewünschten Block, 'W' bedeutet WRITE und schreibt einen Block auf Diskette. Sie brauchen lediglich anzugeben, welchen Track und Sektor Sie lesen wollen. Diese Angaben müssen in

hexadezimaler Form erfolgen, genauso wie auch die Ausgabe auf dem Bildschirm erfolgt. Wenn Sie z.B. Track 18, Sektor 1 lesen wollen (den ersten Directoryblock), geben Sie folgenden Befehl ein:

```
>R 12 01
```

Sämtliche Eingabe müssen also als zweistellige Hexzahlen erfolgen, die durch ein Leerzeichen von einander getrennt sind.

Um sich den Block jetzt auf dem Bildschirm anzusehen, dient der Befehl 'M'. Wir erhalten z.B. folgende Ausgabe:

```
DISK-MONITOR V1.0
>M
>:00  12 04 82 11 01 47 52 41 .....GRA
>:08  46 49 4B 20 41 49 44 2E FIK AID.
>:10  53 52 43 A0 A0 00 00 00 SRC  ...
>:18  00 00 00 00 00 00 15 00 ........
>:20  00 00 82 13 00 48 50 4C .....HPL
>:28  4F 54 2E 53 52 43 A0 A0 OT.SRC
>:30  A0 A0 A0 A0 A0 00 00 00       ...
>:38  00 00 00 00 00 00 05 00 ........
>:40  00 00 82 13 03 56 50 4C .....VPL
>:48  4F 54 2E 53 52 43 A0 A0 OT.SRC
>:50  A0 A0 A0 A0 A0 00 00 00       ...
>:58  00 00 00 00 00 00 09 00 ........
>:60  00 00 82 13 09 4D 45 4D .....MEM
>:68  2E 53 52 43 A0 A0 A0 A0 .SRC
>:70  A0 A0 A0 A0 A0 00 00 00       ...
>:78  00 00 00 00 00 00 06 00 ........
>:80  00 00 82 13 08 4D 45 4D .....MEM
>:88  2E 4F 42 4A A0 A0 A0 A0 .OBJ
>:90  A0 A0 A0 A0 A0 00 00 00       ...
>:98  00 00 00 00 00 00 01 00 ........
>:A0  00 00 82 10 00 53 57 41 .....SWA
>:A8  50 2E 53 52 43 A0 A0 A0 P.SRC
>:B0  A0 A0 A0 A0 A0 00 00 00       ...
>:B8  00 00 00 00 00 00 04 00 ........
>:C0  00 00 82 10 01 4D 41 54 .....MAT
>:C8  52 49 58 2E 53 52 43 A0 RIX.SRC
>:D0  A0 A0 A0 A0 A0 00 00 00       ...
>:D8  00 00 00 00 00 00 0D 00 ........
>:E0  00 00 82 13 0C 47 41 55 .....GAU
>:E8  53 53 2E 54 45 53 54 A0 SS.TEST
>:F0  A0 A0 A0 A0 A0 00 00 00       ...
>:F8  00 00 00 00 00 00 01 00 ........
```

Sehen wir uns die Ausgabe mal etwas genauer an. Die erste Hexzahl nach dem Doppelpunkt gibt die Adresse der folgenden 8 Bytes im Block an, 00 bedeutet das erste Byte innerhalb des Blocks (Die Nummerierung läuft von 00 bis FF bzw. 0 bis 255). Nach der Adresse folgen 8 Bytes (4 auf dem VC 20). In der rechten Hälfte stehen die entsprechenden ASCII-Zeichen.

Handelt es sich um nichtdruckende Zeichen (ASCII-Kode $00 bis $1F und $80 bis $9F), so steht dort ein Punkt. Geben Sie wie oben den Befehl 'M' ein, so wird der ganze Block angezeigt. Da der Block nicht komplett auf den Bildschirm passt, besteht auch die Möglichkeit, sich nur einen Teil anzusehen. Geben Sie dazu den Adressbereich an den Sie anzeigen möchten. Wollen Sie nur die Hälfte sehen, schreiben Sie:

```
>M 00 7F
```

Die zweite Hälfte entsprechend mit:

```
>M 80 FF
```

Beim VC 20 können Sie entsprechend sich jeweils ein Viertel eines Block ansehen. Wollen Sie nun irgendwelche Daten ändern, so gehen Sie einfach mit dem Cursor an die entsprechende Stelle, überschreiben das Byte und drücken Return. Der neue Wert wird jetzt übernommen und gleichzeitig das ASCII-Zeichen in der rechten Hälfte mit geändert.

Wollen Sie nun den veränderten Block wieder auf Diskette zurückschreiben, so benutzen Sie dazu den Befehl 'W'. Auch hierbei müssen Sie wieder hexadezimal angeben, welchen Track und Sektor Sie schreiben wollen.

```
>W 12 01
```

schreibt den Block wieder nach Track 18, Sektor 1, von wo wir den Block vorher gelesen hatten.

Wollen Sie wieder ins BASIC zurück, so geben Sie 'X' ein und der Rechner meldet sich wieder mit 'READY.'. Wollen Sie den Disk-Monitor danach noch einmal benutzen, brauchen Sie ihn nicht mehr neu zu laden, sondern können mit SYS 49152 beim 64er bzw. mit SYS 6690 beim VC 20 wieder in den Monitor springen.

Hier für den Anfang noch eine Warnung:

Machen Sie unbedingt von Ihrer Diskette, die Sie so behandeln wollen, eine Kopie, mit der Sie dann arbeiten. Machen Sie nämlich beim Ändern oder Schreiben eines Blocks einen Fehler, können Sie wichtige Informationen auf der Diskette zerstören und die Diskette ist unter Umständen auf normalem Wege nicht mehr zu lesen. Sie sollten sich es daher zur Regel machen, bei derartigen Manipulation immer mit einer Kopie zu arbeiten.

Nachfolgend finden Sie das Assemblerlisting dieses (etwas längeren) Maschinenprogramms, im Anschluß daran wieder Ladeprogramme in BASIC für den Commodore 64 und den VC 20.

```
                    ; disk monitor  vc20 / cbm 64
                    ;
190:   c000         prompt   =    ">"
200:   c000         ncmds    =    6        ; anzahl der befehle
210:   c000         input    =    $ffcf
220:   c000         talk     =    $ffb4
230:   c000         sectalk  =    $ff96
240:   c000         iecin    =    $ffa5
250:   c000         untalk   =    $ffab
260:   c000         listen   =    $ffb1
270:   c000         seclist  =    $ff93
280:   c000         iecout   =    $ffa8
290:   c000         unlist   =    $ffae
300:   c000         write    =    $ffd2
310:   c000         open     =    $ffc0
320:   c000         close    =    $ffc3
330:   c000         setpar   =    $ffba
340:   c000         setnam   =    $ffbd
350:   c000         chkin    =    $ffc6
360:   c000         ckout    =    $ffc9
370:   c000         clrch    =    $ffcc
380:   c000         cr       =    13
390:   c000         quote    =    $22
400:   c000         quotflg  =    $d4
410:   0200                  *=   $200     ; basic eingabepuffer
420:   0201         savx     *=   *+1
430:   0202         wrap     *=   *+1
440:   0203         bad      *=   *+1
450:   0204         von      *=   *+1
460:   0205         bis      *=   *+1
470:   0205         status   =    $90
480:   0205         sa       =    $b9      ; sekundär adresse
490:   0205         fa       =    $ba      ; gerätenummer
500:   0205         fnadr    =    $bb      ; adresse des filenamens
510:   0205         fnlen    =    $b7      ; länge des filenamens
520:   0205         tmpc     =    $97
610:   c000         count    =    8        ; anzahl der bytes pro zeile
                                           ; 4 beim vc20
620:   c000         ready    =    $e37b    ; $e467 beim vc20
630:   c000 a2 00   init     ldx  #0
640:   c002 bd 85 c2 msgout  lda  message,x
650:   c005 20 d2 ff         jsr  write
660:   c008 e8               inx            ; einschaltmeldung ausgeben
670:   c009 e0 12            cpx  #ascdmp-message
680:   c00b d0 f5            bne  msgout
690:   c00d a2 0d    start   ldx  #cr
700:   c00f a9 3e            lda  #prompt
710:   c011 20 eb c0         jsr  wrtwo
710:   c014 a9 00            lda  #0
710:   c016 8d 01 02         sta  wrap
720:   c019 20 33 c1 stl     jsr  rdoc     ; eingabezeile lesen
730:   c01c c9 3e            cmp  #prompt
740:   c01e f0 f9            beq  stl
750:   c020 c9 20            cmp  #" "     ; leerzeichen überlesen
```

```
760:   c022 f0 f5            beq  stl
770:   c024 a2 05    s0      ldx  #ncmds-1   : mit befehlstabelle vergleichen
780:   c026 dd 6a c0 s1      cmp  cmds,x
790:   c029 d0 0c            bne  s2
800:   c02b 8e 00 02         stx  savx    ; nummer des befehls in der tabelle
840:   c02e bd 70 c0         lda  adrh,x
850:   c031 48               pha          ; rücksprungadresse auf stack
860:   c032 bd 76 c0         lda  adrl,x
870:   c035 48               pha
880:   c036 60               rts
890:   c037 ca       s2      dex
900:   c038 10 ec            bpl  s1      ; schleife über alle befehle
910:   c03a 4c 0d c0         jmp  start
                    ;
                    ; unterprogramm zur anzeige
                    ; des disketteninhalts
960:   c03d 85 97    dm      sta  tmpc
970:   c03f 20 62 c0 dm1     jsr  space
980:   c042 b9 e0 c2         lda  buffer,y  ; byte aus puffer holen
990:   c045 20 dc c0         jsr  wrob
1000:  c048 c8               iny
1000:  c049 d0 03            bne  dm2
1000:  c04b ee 01 02         inc  wrap
1010:  c04e c6 97    dm2     dec  tmpc
1020:  c050 d0 ed            bne  dm1
1030:  c052 60               rts
                    ; bytes lesen und in speicher schreiben
1060:  c053 20 fe c0 byt     jsr  rdob
1070:  c056 90 03            bcc  by3     ; leerzeichen ?
1080:  c058 99 e0 c2         sta  buffer,y  ; byte in puffer schreiben
1090:  c05b c8       by3     iny
1100:  c05c c6 97            dec  tmpc
1110:  c05e 60               rts
1120:  c05f 20 62 c0 spac2   jsr  space
1130:  c062 a9 20    space   lda  #" "
1140:  c064 2c               .byte $2c
1150:  c065 a9 0d    crlf    lda  #cr
1160:  c067 4c d2 ff         jmp  write
                    ;
                    ; befehls- und adresstabelle
1190:  c06a 3a       cmds    .byt ":"     ; speicherinhalt ändern
1200:  c06b 57               .byt "w"     ; block schreiben
1210:  c06c 52               .byt "r"     ; block lesen
1220:  c06d 4d               .byt "m"     ; bytes anzeigen
1230:  c06e 40               .byt "@"     ; disketten befehl
1240:  c06f 58               .byt "x"     ; exit
1250:  c070 c0       adrh    .byt >altm-1
1260:  c071 c1               .byt >direkt-1
1270:  c072 c1               .byt >direkt-1
1280:  c073 c0               .byt >dsplym-1
1290:  c074 c1               .byt >disk-1
1300:  c075 e3               .byt >ready-1
1310:  c076 c0       adrl    .byt <altm-1
1320:  c077 90               .byt <direkt-1
1330:  c078 90               .byt <direkt-1
```

```
1340:  c079 7b                .byt <dsplym-1
1350:  c07a 3e                .byt <disk-1
1360:  c07b 7a                .byt <ready-1
1370:  c07c a0 00    dsplym   ldy  #0
1370:  c07e 8c 03 02          sty  von
1370:  c081 88                dey
1370:  c082 8c 04 02          sty  bis
1370:  c085 20 cf ff          jsr  input
1370:  c088 c9 0d             cmp  #cr
1370:  c08a f0 17             beq  dspl
1380:  c08c 20 fe c0          jsr  rdob     ; startadresse lesen
1390:  c08f 90 12             bcc  dspl
1400:  c091 8d 03 02          sta  von
1410:  c094 20 cf ff          jsr  input
1410:  c097 c9 0d             cmp  #cr
1410:  c099 f0 08             beq  dspl
1420:  c09b 20 fe c0          jsr  rdob     ; endadresse lesen
1430:  c09e 90 03             bcc  dspl
1440:  c0a0 8d 04 02          sta  bis
1450:  c0a3 ac 03 02 dspl     ldy  von
1460:  c0a6 20 c6 c2 dsp2     jsr  testend
1470:  c0a9 20 d6 c2          jsr  altrit
1470:  c0ac 98                tya
1480:  c0ad 20 dc c0          jsr  wrob     ; adresse
1490:  c0b0 20 62 c0          jsr  space      ; beim vc 20 weglassen
1500:  c0b3 a9 08             lda  #count   ; 8 oder 4
1510:  c0b5 20 3d c0          jsr  dm       ; anzeigen
1520:  c0b8 20 97 c2          jsr  ascdmp   ; ascii-dump
1530:  c0bb 4c a6 c0          jmp  dsp2     ; unbedingter sprung
1550:  c0be 4c 0d c0 beqsl    jmp  start
                        ;speicher ändern ; adresse und daten lesen
1570:  c0c1 20 fe c0 altm     jsr  rdob     ; adresse lesen
1580:  c0c4 90 f8             bcc  beqsl    ;
1590:  c0c6 a8                tay
1600:  c0c7 a9 08             lda  #count   ; anzahl der bytes
1610:  c0c9 85 97             sta  tmpc
1610:  c0cb 20 33 c1          jsr  rdoc       ; beim vc20 weglassen
1620:  c0ce 20 33 c1 a5       jsr  rdoc
1620:  c0d1 20 53 c0          jsr  byt
1630:  c0d4 d0 f8             bne  a5
1640:  c0d6 20 97 c2          jsr  ascdmp
1650:  c0d9 4c 0d c0          jmp  start
                        ;
                        ; byte als hexzahl schreiben
1710:  c0dc 48          wrob  pha
1720:  c0dd 4a                lsr  a
1730:  c0de 4a                lsr  a
1740:  c0df 4a                lsr  a
1750:  c0e0 4a                lsr  a
1760:  c0e1 20 f4 c0          jsr  ascii    ; nach ascii convertieren
1770:  c0e4 aa                tax
1780:  c0e5 68                pla
1790:  c0e6 29 0f             and  #%1111
1800:  c0e8 20 f4 c0          jsr  ascii
                        ; zeichen in x und a schreiben
```

```
1820:  c0eb 48       wrtwo    pha
1830:  c0ec 8a                txa
1840:  c0ed 20 d2 ff          jsr  write
1850:  c0f0 68                pla
1860:  c0f1 4c d2 ff          jmp  write
1870:  c0f4 18       ascii    clc
1880:  c0f5 69 f6             adc  #$f6
1890:  c0f7 90 02             bcc  ascl
1900:  c0f9 69 06             adc  #6
1910:  c0fb 69 3a    ascl     adc  #$3a
1920:  c0fd 60                rts
                    ; hexbyte lesen und nach a bringen
1950:  c0fe a9 00    rdob     lda  #0
1960:  c100 8d 02 02          sta  bad      ; nächstes zeichen lesen
1970:  c103 20 33 c1          jsr  rdoc
1980:  c106 c9 20    rdobl    cmp  #" "
1990:  c108 d0 09             bne  rdob2
2000:  c10a 20 33 c1          jsr  rdoc     ; nächstes zeichen lesen
2010:  c10d c9 20             cmp  #" "
2020:  c10f d0 0f             bne  rdob3
2030:  c111 18                clc           ; cy=0
2040:  c112 60                rts
2050:  c113 20 28 c1 rdob2    jsr  hexit    ; nach hex wandeln
2060:  c116 0a                asl  a
2070:  c117 0a                asl  a
2080:  c118 0a                asl  a
2090:  c119 0a                asl  a
2100:  c11a 8d 02 02          sta  bad
2110:  c11d 20 33 c1          jsr  rdoc
2120:  c120 20 28 c1 rdob3    jsr  hexit
2130:  c123 0d 02 02          ora  bad
2140:  c126 38                sec           ; cy=1
2150:  c127 60                rts
2160:  c128 c9 3a    hexit    cmp  #$3a
2170:  c12a 08                php
2180:  c12b 29 0f             and  #%1111
2190:  c12d 28                plp
2200:  c12e 90 02             bcc  hex09    ; 0-9
2210:  c130 69 08             adc  #8       ; plus 9 (c-1)
2220:  c132 60       hex09    rts
2230:  c133 20 cf ff rdoc     jsr  input    ; zeichen lesen
2240:  c136 c9 0d             cmp  #cr      ; cr ?
2250:  c138 d0 f8             bne  hex09    ; nein, return
2260:  c13a 68                pla
2270:  c13b 68                pla           ; ja, zum start
2280:  c13c 4c 0d c0          jmp  start
                    ;
                    ; dos support
2320:  c13f 20 cf ff disk     jsr  input
2330:  c142 c9 0d             cmp  #cr
2340:  c144 d0 27             bne  dskcmd   ; disk command
2350:  c146 a9 00             lda  #0
2350:  c148 85 90             sta  status   ; status löschen
2360:  c14a 20 65 c0          jsr  crlf
2370:  c14d a9 08             lda  #8
```

```
2380:  c14f 85 ba             sta  fa      ; floppyadresse
2390:  c151 20 b4 ff          jsr  talk
2400:  c154 a9 6f             lda  #15+$60  ; sa 15
2410:  c156 85 b9             sta  sa
2420:  c158 20 96 ff          jsr  sectalk  ; sekadr
2430:  c15b 20 a5 ff errin    jsr  iecin
2440:  c15e 24 90             bit  status
2440:  c160 70 05             bvs  enddsk
2450:  c162 20 d2 ff          jsr  write
2460:  c165 d0 f4             bne  errin
2470:  c167 20 ab ff enddsk   jsr  untalk
2480:  c16a 4c 0d c0          jmp  start
2490:  c16d c9 24    dskcmd   cmp  #"$"
2500:  c16f f0 1d             beq  errl    ; catalog
2510:  c171 48                pha
2510:  c172 a9 08             lda  #8
2520:  c174 85 ba             sta  fa
2530:  c176 20 b1 ff          jsr  listen
2540:  c179 a9 6f             lda  #15+$60
2550:  c17b 85 b9             sta  sa
2560:  c17d 20 93 ff          jsr  seclist
2560:  c180 68                pla
2570:  c181 20 a8 ff cmdout   jsr  iecout
2580:  c184 20 cf ff          jsr  input
2590:  c187 c9 0d             cmp  #cr
2600:  c189 d0 f6             bne  cmdout
2610:  c18b 20 ae ff          jsr  unlist
2630:  c18e 4c 0d c0 errl     jmp  start
2640:  c191 20 33 c1 direkt   jsr  rdoc
2640:  c194 20 fe c0          jsr  rdob    ; track lesen
2650:  c197 90 f5             bcc  errl
2660:  c199 8d 27 c2          sta  track
2670:  c19c 20 33 c1          jsr  rdoc
2670:  c19f 20 fe c0          jsr  rdob
2680:  c1a2 90 ea             bcc  errl
2690:  c1a4 8d 2a c2          sta  sector
2690:  c1a7 20 49 c2          jsr  opndir
2690:  c1aa ad 00 02          lda  savx
2690:  c1ad c9 01             cmp  #1
2690:  c1af f0 1e             beq  dirwrite
2700:  c1b1 a9 31             lda  #"1"
2710:  c1b3 20 ed c1          jsr  sendcmd  ; block-read befehl senden
2720:  c1b6 a2 0d             ldx  #13
2730:  c1b8 20 c6 ff          jsr  chkin
2740:  c1bb a2 00             ldx  #0
2750:  c1bd 20 cf ff dirin    jsr  input
2760:  c1c0 9d e0 c2          sta  buffer,x
2770:  c1c3 e8                inx
2770:  c1c4 d0 f7             bne  dirin
2780:  c1c6 20 cc ff          jsr  clrch
2790:  c1c9 20 6e c2 enddir   jsr  clsdir
2790:  c1cc 4c 0d c0          jmp  start
2800:  c1cf 20 2c c2 dirwrite jsr  bufpnt  ; bufferpointer setzen
2810:  c1d2 a2 0d             ldx  #13
2820:  c1d4 20 c9 ff          jsr  ckout
```

```
2830:  cld7 a2 00             ldx  #0
2840:  cld9 bd e0 c2 dirout   lda  buffer,x
2850:  cldc 20 d2 ff          jsr  write
2860:  cldf e8                inx
2860:  cle0 d0 f7             bne  dirout
2870:  cle2 20 cc ff          jsr  clrch
2880:  cle5 a9 32             lda  #"2"
2890:  cle7 20 ed cl          jsr  sendcmd  ; block-write befehl senden
2900:  clea 4c c9 cl          jmp  enddir
2910:  cled 8d 20 c2 sendcmd  sta  cmdstr+1
2910:  clf0 a2 0f             ldx  #15
2920:  clf2 ad 27 c2          lda  track
2920:  clf5 20 78 c2          jsr  numbasc
2920:  clf8 8e 27 c2          stx  track
2920:  clfb 8d 28 c2          sta  track+1
2930:  clfe ad 2a c2          lda  sector
2930:  c201 20 78 c2          jsr  numbasc
2930:  c204 8e 2a c2          stx  sector
2930:  c207 8d 2b c2          sta  sector+1
2940:  c20a a2 0f             ldx  #15
2940:  c20c 20 c9 ff          jsr  ckout
2950:  c20f a2 00             ldx  #0
2960:  c211 bd 1f c2 comdout  lda  cmdstr,x
2970:  c214 20 d2 ff          jsr  write
2980:  c217 e8                inx
2980:  c218 e0 0d             cpx  #bufpnt-cmdstr
2990:  c21a d0 f5             bne  comdout
3000:  c21c 4c cc ff          jmp  clrch
3010:  c21f 55 31 3a cmdstr   .asc "ul:13 0 "
3020:  c227 00 00 20 track    .byt 0,0, " "
3030:  c22a 00 00    sector   .byt 0,0
3040:  c22c a2 0f    bufpnt   ldx  #15
3050:  c22e 20 c9 ff          jsr  ckout
3060:  c231 a2 00             ldx  #0
3070:  c233 bd 41 c2 pntout   lda  buftxt,x
3080:  c236 20 d2 ff          jsr  write
3090:  c239 e8                inx
3090:  c23a e0 08             cpx  #opndir-buftxt
3100:  c23c d0 f5             bne  pntout
3110:  c23e 4c cc ff          jmp  clrch
3120:  c241 42 2d 50 buftxt   .asc "b-p 13 0"
3130:  c249 a9 0f    opndir   lda  #15
3130:  c24b a8                tay
3140:  c24c a2 08             ldx  #8
3150:  c24e 20 ba ff          jsr  setpar
3160:  c251 a9 00             lda  #0
3170:  c253 20 bd ff          jsr  setnam
3180:  c256 20 c0 ff          jsr  open
3190:  c259 a9 0d             lda  #13
3190:  c25b a8                tay
3200:  c25c a2 08             ldx  #8
3210:  c25e 20 ba ff          jsr  setpar
3220:  c261 a9 01             lda  #1
3230:  c263 a2 6d             ldx  #< dadr
3240:  c265 a0 c2             ldy  #> dadr
```

```
3250:  c267 20 bd ff          jsr  setnam
3260:  c26a 4c c0 ff          jmp  open
3270:  c26d 23        dadr    .byt "#"
3280:  c26e a9 0d     clsdir  lda  #13
3290:  c270 20 c3 ff          jsr  close
3300:  c273 a9 0f             lda  #15
3310:  c275 4c c3 ff          jmp  close
3320:  c278 a2 30     numbasc ldx  #"0"    ; hexzahl nach ascii
3330:  c27a 38                sec
3340:  c27b e9 0a     numbl   sbc  #10
3350:  c27d 90 03             bcc  numb2
3360:  c27f e8                inx
3370:  c280 b0 f9             bcs  numbl
3380:  c282 69 3a     numb2   adc  #"9"+1
3390:  c284 60                rts
3400:  c285 0d        message .byt cr
3410:  c286 44 49 53          .asc "disk-monitor v1.0"
3430:  c297 98        ascdmp  tya
3440:  c298 38                sec
3440:  c299 e9 08             sbc  #count
3440:  c29b a8                tay
3450:  c29c 20 62 c0          jsr  space
3460:  c29f a9 12             lda  #18     ; rvs on
3470:  c2a1 20 d2 ff          jsr  write
3480:  c2a4 a2 08             ldx  #count
3490:  c2a6 b9 e0 c2 ac2      lda  buffer,y
3500:  c2a9 29 7f             and  #$7f
3510:  c2ab c9 20             cmp  #" "
3520:  c2ad b0 04             bcs  ac3
3530:  c2af a9 2e             lda  #"."
3540:  c2b1 d0 03             bne  ac4
3550:  c2b3 b9 e0 c2 ac3      lda  buffer,y
3560:  c2b6 20 d2 ff ac4      jsr  write
3570:  c2b9 a9 00             lda  #0
3570:  c2bb 85 d4             sta  quotflg
3580:  c2bd c8                iny
3580:  c2be ca                dex
3590:  c2bf d0 e5             bne  ac2
3600:  c2c1 a9 92             lda  #146    ; rvs off
3610:  c2c3 4c d2 ff          jmp  write
3620:  c2c6 ad 01 02 testend  lda  wrap
3620:  c2c9 d0 06             bne  endend
3630:  c2cb cc 04 02          cpy  bis
3640:  c2ce b0 01             bcs  endend
3650:  c2d0 60                rts
3660:  c2d1 68        endend  pla
3660:  c2d2 68                pla
3660:  c2d3 4c 0d c0          jmp  start
3670:  c2d6 20 65 c0 altrit   jsr  crlf
3680:  c2d9 a9 3a             lda  #":"
3690:  c2db a2 3e             ldx  #prompt
3700:  c2dd 4c eb c0          jmp  wrtwo
3730:  c2e0           buffer  =    *       ; 256 bytes buffer für block
```

Nachfolgend finden Sie wieder das BASIC-Programm zur Eingabe des Disk-Monitors.

Disk-Monitor, 64er Version

```
100 for i = 49152 to 49887
110 read x : poke i,x : s=s+x : next
120 data 162,  0,189,133,194, 32,210,255,232,224, 18,208
130 data 245,162, 13,169, 62, 32,235,192,169,  0,141,  1
140 data   2, 32, 51,193,201, 62,240,249,201, 32,240,245
150 data 162,  5,221,106,192,208, 12,142,  0,  2,189,112
160 data 192, 72,189,118,192, 72, 96,202, 16,236, 76, 13
170 data 192,133,151, 32, 98,192,185,224,194, 32,220,192
180 data 200,208,  3,238,  1,  2,198,151,208,237, 96, 32
190 data 254,192,144,  3,153,224,194,200,198,151, 96, 32
200 data  98,192,169, 32, 44,169, 13, 76,210,255, 58, 87
210 data  82, 77, 64, 88,192,193,193,192,193,227,192,144
220 data 144,123, 62,122,160,  0,140,  3,  2,136,140,  4
230 data   2, 32,207,255,201, 13,240, 23, 32,254,192,144
240 data  18,141,  3,  2, 32,207,255,201, 13,240,  8, 32
250 data 254,192,144,  3,141,  4,  2,172,  3,  2, 32,198
260 data 194, 32,214,194,152, 32,220,192, 32, 98,192,169
270 data   8, 32, 61,192, 32,151,194, 76,166,192, 76, 13
280 data 192, 32,254,192,144,248,168,169,  8,133,151, 32
290 data  51,193, 32, 51,193, 32, 83,192,208,248, 32,151
300 data 194, 76, 13,192, 72, 74, 74, 74, 74, 32,244,192
310 data 170,104, 41, 15, 32,244,192, 72,138, 32,210,255
320 data 104, 76,210,255, 24,105,246,144,  2,105,  6,105
330 data  58, 96,169,  0,141,  2,  2, 32, 51,193,201, 32
340 data 208,  9, 32, 51,193,201, 32,208, 15, 24, 96, 32
350 data  40,193, 10, 10, 10, 10,141,  2,  2, 32, 51,193
360 data  32, 40,193, 13,  2,  2, 56, 96,201, 58,  8, 41
370 data  15, 40,144,  2,105,  8, 96, 32,207,255,201, 13
380 data 208,248,104,104, 76, 13,192, 32,207,255,201, 13
390 data 208, 39,169,  0,133,144, 32,101,192,169,  8,133
400 data 186, 32,180,255,169,111,133,185, 32,150,255, 32
410 data 165,255, 36,144,112,  5, 32,210,255,208,244, 32
420 data 171,255, 76, 13,192,201, 36,240, 29, 72,169,  8
430 data 133,186, 32,177,255,169,111,133,185, 32,147,255
440 data 104, 32,168,255, 32,207,255,201, 13,208,246, 32
450 data 174,255, 76, 13,192, 32, 51,193, 32,254,192,144
460 data 245,141, 39,194, 32, 51,193, 32,254,192,144,234
470 data 141, 42,194, 32, 73,194,173,  0,  2,201,  1,240
480 data  30,169, 49, 32,237,193,162, 13, 32,198,255,162
490 data   0, 32,207,255,157,224,194,232,208,247, 32,204
500 data 255, 32,110,194, 76, 13,192, 32, 44,194,162, 13
510 data  32,201,255,162,  0,189,224,194, 32,210,255,232
520 data 208,247, 32,204,255,169, 50, 32,237,193, 76,201
530 data 193,141, 32,194,162, 15,173, 39,194, 32,120,194
540 data 142, 39,194,141, 40,194,173, 42,194, 32,120,194
550 data 142, 42,194,141, 43,194,162, 15, 32,201,255,162
560 data   0,189, 31,194, 32,210,255,232,224, 13,208,245
570 data  76,204,255, 85, 49, 58, 49, 51, 32, 48, 32,  0
```

```
580 data   0, 32,  0,  0,162, 15, 32,201,255,162,  0,189
590 data  65,194, 32,210,255,232,224,  8,208,245, 76,204
600 data 255, 66, 45, 80, 32, 49, 51, 32, 48,169, 15,168
610 data 162,  8, 32,186,255,169,  0, 32,189,255, 32,192
620 data 255,169, 13,168,162,  8, 32,186,255,169,  1,162
630 data 109,160,194, 32,189,255, 76,192,255, 35,169, 13
640 data  32,195,255,169, 15, 76,195,255,162, 48, 56,233
650 data  10,144,  3,232,176,249,105, 58, 96, 13, 68, 73
660 data  83, 75, 45, 77, 79, 78, 73, 84, 79, 82, 32, 86
670 data  49, 46, 48,152, 56,233,  8,168, 32, 98,192,169
680 data  18, 32,210,255,162,  8,185,224,194, 41,127,201
690 data  32,176,  4,169, 46,208,  3,185,224,194, 32,210
700 data 255,169,  0,133,212,200,202,208,229,169,146, 76
710 data 210,255,173,  1,  2,208,  6,204,  4,  2,176,  1
720 data  96,104,104, 76, 13,192, 32,101,192,169, 58,162
730 data  62, 76,235,192
740 if s <> 90444 then print "fehler in datas !!" : end
750 sys 49152
```

Disk-Monitor, 20er Version

Damit das Programm auch auf dem VC 20 in der Grundversion läuft, wurde das Ladeprogramm in zwei Teile zerlegt. Geben Sie beide Programm ein und speichern Sie sie jeweils unter dem Namen "dos lader.1" bzw. "dos lader.2" auf Diskette ab. Um den DOS-Monitor zu laden, laden Sie bitte das erste Programm ('dos lader.1') von Diskette und starten Sie es mit 'run'. Wenn alle data's in Ordnung sind, wird automatisch der zweite Teil des Laderprogramms nachgeladen und anschließend der DOS-Monitor gestartet, sofern auch hier keine Fehler in den data-Statements sind.

```
100 poke 55, 6690 and 255 : poke 56, 6690 / 256 : clr
105 for i = 6690 to 7056  :rem dos lader.1
110 read x : poke i,x : s=s+x : next
120 data 162,  0,189,164, 28, 32,210,255,232,224, 18,208
130 data 245,162, 13,169, 62, 32,  7, 27,169,  0,141,  1
140 data   2, 32, 79, 27,201, 62,240,249,201, 32,240,245
150 data 162,  5,221,140, 26,208, 12,142,  0,  2,189,146
160 data  26, 72,189,152, 26, 72, 96,202, 16,236, 76, 47
170 data  26,133,151, 32,132, 26,185,  0, 29, 32,248, 26
180 data 200,208,  3,238,  1,  2,198,151,208,237, 96, 32
190 data  26, 27,144,  3,153,  0, 29,200,198,151, 96, 32
200 data 132, 26,169, 32, 44,169, 13, 76,210,255, 58, 87
210 data  82, 77, 64, 88, 26, 27, 27, 26, 27,228,223,175
220 data 175,157, 90,102,160,  0,140,  3,  2,136,140,  4
230 data   2, 32,207,255,201, 13,240, 23, 32, 26, 27,144
240 data  18,141,  3,  2, 32,207,255,201, 13,240,  8, 32
250 data  26, 27,144,  3,141,  4,  2,172,  3,  2, 32,229
260 data  28, 32,245, 28,152, 32,248, 26,169,  4, 32, 95
270 data  26, 32,182, 28, 76,200, 26, 76, 47, 26, 32, 26
280 data  27,144,248,168,169,  4,133,151, 32, 79, 27, 32
290 data 117, 26,208,248, 32,182, 28, 76, 47, 26, 72, 74
```

```
300 data  74, 74, 74, 32, 16, 27,170,104, 41, 15, 32, 16
310 data  27, 72,138, 32,210,255,104, 76,210,255, 24,105
320 data 246,144,  2,105,  6,105, 58, 96,169,  0,141,  2
330 data   2, 32, 79, 27,201, 32,208,  9, 32, 79, 27,201
340 data  32,208, 15, 24, 96, 32, 68, 27, 10, 10, 10, 10
350 data 141,  2,  2, 32, 79, 27, 32, 68, 27, 13,  2,  2
360 data  56, 96,201, 58,  8, 41, 15, 40,144,  2,105,  8
370 data  96, 32,207,255,201, 13,208,248,104,104, 76, 47
380 data  26, 32,207,255,201, 13,208, 39,169,  0,133,144
390 data  32,135, 26,169,  8,133,186, 32,180,255,169,111
400 data 133,185, 32,150,255, 32,165,255, 36,144,112,  5
410 data  32,210,255,208,244, 32,171,255, 76, 47, 26,201
420 data  36,240, 29, 72,169,  8,133
430 if s <> 35614 then print "fehler in datas !!" : end
440 load "dos lader.2",8


100 clr : for i = 7057 to 7422 :rem dos lader.2
110 read x : poke i,x : s=s+x : next
120 data 186, 32,177,255,169,111,133,185, 32,147,255,104
130 data  32,168,255, 32,207,255,201, 13,208,246, 32,174
140 data 255, 76, 47, 26, 76, 47, 26, 32, 79, 27, 32, 26
150 data  27,144,245,141, 70, 28, 32, 79, 27, 32, 26, 27
160 data 144,234,141, 73, 28, 32,104, 28,173,  0,  2,201
170 data   1,240, 30,169, 49, 32, 12, 28,162, 13, 32,198
180 data 255,162,  0, 32,207,255,157,  0, 29,232,208,247
190 data  32,204,255, 32,141, 28, 76, 47, 26, 32, 75, 28
200 data 162, 13, 32,201,255,162,  0,189,  0, 29, 32,210
210 data 255,232,208,247, 32,204,255,169, 50, 32, 12, 28
220 data  76,232, 27,141, 63, 28,162, 15,173, 70, 28, 32
230 data 151, 28,142, 70, 28,141, 71, 28,173, 73, 28, 32
240 data 151, 28,142, 73, 28,141, 74, 28,162, 15, 32,201
250 data 255,162,  0,189, 62, 28, 32,210,255,232,224, 13
260 data 208,245, 76,204,255, 85, 49, 58, 49, 51, 32, 48
270 data  32,  0,  0, 32,  0,  0,162, 15, 32,201,255,162
280 data   0,189, 96, 28, 32,210,255,232,224,  8,208,245
290 data  76,204,255, 66, 45, 80, 32, 49, 51, 32, 48,169
300 data  15,168,162,  8, 32,186,255,169,  0, 32,189,255
310 data  32,192,255,169, 13,168,162,  8, 32,186,255,169
320 data   1,162,140,160, 28, 32,189,255, 76,192,255, 35
330 data 169, 13, 32,195,255,169, 15, 76,195,255,162, 48
340 data  56,233, 10,144,  3,232,176,249,105, 58, 96, 13
350 data  68, 73, 83, 75, 45, 77, 79, 78, 73, 84, 79, 82
360 data  32, 86, 49, 46, 48,152, 56,233,  4,168, 32,132
370 data  26,169, 18, 32,210,255,162,  4,185,  0, 29, 41
380 data 127,201, 32,176,  4,169, 46,208,  3,185,  0, 29
390 data  32,210,255,169,  0,133,212,200,202,208,229,169
400 data 146, 76,210,255,173,  1,  2,208,  6,204,  4,  2
410 data 176,  1, 96,104,104, 76, 47, 26, 32,135, 26,169
420 data  58,162, 62, 76,  7, 27
430 if s <> 39496 then print "fehler in datas !!" : end
440 sys 6690
```

5 Die großen CBM-Floppys

5.1 IEC-Bus und serieller Bus

Commodore 64 und VC 20 haben serienmäßig einen seriellen Bus, über den Peripheriegeräte angeschlossen werden können, z.B. die Floppy VC 1541 sowie Drucker und Plotter.

Das Busprinzip ermöglicht es, die Geräte gleichzeitig anzuschließen. Damit die Gräte unterschieden werden können, wird jedem Gerät eine Geräteadresse zugewiesen, unter der man das Gerät ansprechen kann. Die Standardadresse der Floppy ist 8, ein Drucker wird meist mit Adresse 4 angesprochen. Die Geräteadresse ist identisch mit der Primäradresse im OPEN-Befehl, so öffnet z.B.

```
OPEN 1,4
```

einen Kanal zum Drucker. Um bei der Floppy nun mehrere Dateien gleichzeitig öffnen zu können, dient eine weitere Adresse, die Sekundäradresse, zur Unterscheidung. Die Floppy verfügt über 16 Sekundäradressen von 0 bis 15. Drei Sekundäradressen dienen festen Zwecken, während die übrigen 13 frei benutzt werden können:

Sekundäradresse 0 dient zum Laden von Programmen.

Sekundäradresse 1 dient zum Abspeichern von Programmen.

Sekundäradresse 15 ist der Kommando- und Fehlerkanal.

Die übrigen Sekundäradressen 2 bis 14 können frei zum Öffnen von Dateien benutzt werden.

Die Übertragung zwischen Commodore 64 und VC 1541 geschieht seriell über diesen Bus. Dabei bedeutet seriell, daß die Daten bitweise über nur eine Leitung übertragen werden. Intern werden die Daten im Rechner und Floppy jeweils zu 8 Bit gleich ein Byte gleichzeitig gespeichert und verarbeitet. Soll ein Byte nun seriell übertragen werden, so wird jedes Bit einzeln über eine Datenleitung gesandt. Damit Sender und Empfänger sich bei der Übertragung auf einander abstimmen können, werden noch sogenannte 'Handshake'-Leitungen benötigt. Sehen wir uns den Anschluß des seriellen Bus einmal genauer an, so finden wir 6 Leitungen:

Pin	Belegung
1	SRQ IN
2	Masse
3	ATN
4	CLCK
5	DATA
6	RESET

Will der Rechner Daten zur Floppy übertragen, so wird die Leitung ATN (Attention, Achtung) gesetzt. Ist dieses Signal gesetzt, unterbrechen alle Geräte am Bus ihre augenblickliche Arbeit und übernehmen das nachfolgend übertragene Byte. Die Daten kommen bitweise über die Leitung DATA. Damit die Empfänger wissen, wann das nächste Bit kommt, wird dies über die Leitung CLCK (Clock, Takt) signalisiert. Dieses übertragene Byte ist die Geräteadresse. Stimmt dieser Wert nicht mit der Geräteadresse der empfangenden Geräts überein, werden die weiteren Daten ignoriert. Ist das Gerät jedoch adressiert, so kann eine evtl. Sekundäradresse übertragen werden. Gleichzeitig mit der Geräteadresse (0 bis 31) wurde mittels der restlichen drei Bit dem Gerät noch mitgeteilt, ob es Daten empfangen (LISTEN) oder selbst Daten senden (TALK) soll. Abhängig davon werden jetzt Daten vom Rechner oder von adressierten Gerät gesandt.

Die Leitung RESET versetzt beim Einschalten des Computers alle angschlossenen Geräte in den Grundzustand. Über die Leitung SRQ IN (Service Request, Bedienungsanforderung) können Peripheriegeräte dem Buscontroller (in unserem Falle immmer dem Computer) melden, wenn z.B. Daten bereit stehen. Diese Leitung wird jedoch vom Betriebssystem der Commodorerechner nicht abgefragt.

Will man mehrere Floppys gleichzeitig anschließen, so müssen die Geräte unterschiedliche Adressen haben. Soll dies nur gelegentlich geschehen, kann dies mit dem Programm 'DISK ADR CHANGE' geschehen, das in Abschnitt 4.2.3 beschrieben ist. Die neue Adresse, z.B. 9, bleibt jedoch nur solange erhalten, bis das Gerät wieder ausgeschaltet wird. Soll die Änderung dauerhaft sein, kann dies durch Trennen einer Brücke im Gerät erfolgen.

Analog zum Prinzip des seriellen Bus funktioniert auch die Datenübertragung über den IEC- oder IEEE 488 Bus. Der wichtigste Unterschied besteht jedoch darin, daß die Daten nicht seriell, sondern parallel über 8 Datenleitungen gleichzeitig übertragen werden. Außerdem sind noch zusätzliche Handshakeleitungen vorhanden, so daß der parallele IEC-Bus ein 24adriges Kabel benötigt. Der Hauptvorteil des IEEE 488 Bus besteht aufgrund der gleichzeitigen Übertragung eines kompletten Bytes in dem damit erreichten Geschwindigkeitsvorteil. Durch Messungen ergibt sich, daß der IEC-Bus etwa 5 mal schneller als der serielle Bus ist: 1.8 KB/s gegenüber 0.4 KB/s. Damit dauert das Laden eines Programms von 10 KByte mit der VC 1541 ca. 25 Sekunden; auf der sonst identischen CBM 2031 jedoch weniger als 6 Sekunden. Allein aus diesem Grunde kann es sich also schon lohnen, seinen Rechner mit einem IEC-Bus auszurüsten.

Gleichzeitig besteht damit die Möglichkeit, auf alle anderen Peripheriegeräte der großen CBM-Computer zugreifen zu können.

5.2 Gegenüberstellung aller CBM-Floppy

In der folgenden Tabelle finden Sie die technischen Daten aller CBM-Floppys zum Vergleich gegenübergestellt.

Die technischen Daten aller Commodore-Floppy-Laufwerke

Modell	1541	2031	4040	8050	8250
DOS-Version(en)	2.6	2.6	2.1/ 2.7	2.5/ 2.7	2.7
Laufwerke	1	1	2	2	2
Köpfe pro Laufwerk	1	1	1	1	2
Speicherkapazität	170 K	170 K	340 K	1.05 M	2.12 M
Sequentielle Datei	168 K	168 K	168 K	521 K	1.05 M
Relative Datei	167 K	167 K	167 K	183 K/ 518 K	1.04 M
Pufferspeicher (KB)	2	2	4	4	4
Tracks	35	35	35	77	77
Sektoren pro Track	17-21	17-21	17-21	23-29	23-29
Bytes pro Block	256	256	256	256	256
freie Blocks	664	664	1328	4104	8266
Directory und BAM (Track)	18	18	18	38/39	38/39
Directoryeinträge	144	144	144	224	224
Übertragungsrate (KB/s)					
intern	40	40	40	40	40
über ser./IEC-Bus	0.4	1.8	1.8	1.8	1.8
Zugriffszeiten (ms)					
Track zu Track	30	30	30	5	5
mittlere Zeit	360	360	360	125	125
Umdrehungen pro Minute	300	300	300	300	300

Überblick über die "großen" CBM-Floppys

Die VC-1541 Floppy ist von der Speicherkapazität her die kleinste CBM-Floppy, bis jetzt jedoch auch die einzige Floppy mit seriellem Bus zum direkten Anschluß an Commodore 64 und VC 20.

Von den Funktionen, dem Aufbau und der Arbeitsweise her identisch ist die Floppy CBM 2031. Der einzige Unterschied zur VC 1541 ist die Ausrüstung mit dem parallelen IEEE 488 Bus im Gegensatz zum seriellen Bus. Dies bringt eine bedeutende Erhöhung der Übertragungsgeschwindigkeit zum

Rechner etwa um den Faktor 5 mit sich. Zum Anschluß an Commodore 64 oder VC 20 benötigt man ein IEC-Bus-Modul, ebenso wie bei allen weiteren CBM-Floppys. Vom Speicherformat ist die CBM 2031 voll kompatibel zur VC 1541; beide haben 170 KB pro Diskette. Disketten die auf einem Gerät beschrieben wurden, können vom jeweils anderen Gerät gelesen und gschrieben werden. Dies gilt auch für die nächste Floppy in dieser Reihe, die CBM 4040. Die 4040 ist ein Doppellaufwerk mit zweimal 170 KB.

Der Vorteil eines Doppellaufwerks liegt nicht allein in der doppelten Speicherkapazität, sondern vor allem in der Möglichkeit, Daten von einem Laufwerk zum anderen zu übertragen. Dies ist einmal mit kompletten Programmen und Dateien mit dem auch bei der 1541 vorhandenen Befehl 'copy' möglich, z.B. kopiert

```
OPEN 1,8,15, "C1:TEST=0:TEST"  bzw.

COPY "TEST",D0 TO "TEST",D1
```

die Datei 'TEST' von Laufwerk 0 unter dem gleichen Namen auf Laufwerk 1. Ebenso kann man mehrere Dateien von unterschiedlichen Laufwerken zusammenfügen ('concat'). Die wichtigste Möglichkeit des Doppellaufwerks ist jedoch das Duplizieren von kompletten Disketten. Dies geschieht ebenfalls mit einem Befehl vom Rechner; das Laufwerk formatiert dann automatisch die neue Diskette und kopiert dann Track für Track von einem Laufwerk auf das andere. Der Befehl dazu lautet:

```
OPEN 1,8,15, "D1=0"   bzw.

BACKUP D0 TO D1
```

Das ganze dauert auf der 4040 keine 3 Minuten; der Rechner kann während dieser Zeit weiterarbeiten, da die Floppy diese Arbeit komplett übernimmt.

Die beiden anderen CBM-Floppys CBM 8050 und 8250 beschreiben die Disketten mit doppelter Dichte ('double density', 77 Tracks). Auf der 1541 bzw. 4040 beschriebene Disketten sind dem zufolge nicht mit 8050/8250-Disketten kompatibel. Programme und Daten lassen sich jedoch, z.B. mit dem Programm 'COPY/ALL', von einem Format auf ein anderes übertragen. Dafür treten diese Flopppies durch die bedeutend höhere Speicherkapazität hervor: 1 MB bei der 8050 und 2 MB bei der 8250. Die doppelte Kapazität der 8250 wird durch Ausnutzen beider Diskettenseiten von der Floppy erreicht ('double sided'), sie hat 2 Schreib/Leseköpfe pro Laufwerk. Um die gesamte Kapazität auch für relative Files ausnutzen zu können (siehe Kapitel 3.4) wurde hierbei ein sogenannter 'Super-Side-Sektor' eingeführt, der die Zeiger auf 127 Gruppen von je 6 Side-Sektor-Blöcken enthält. Dadurch kann hier eine relative Datei (theoretisch) 23 MB umfassen (bei der 8050 ab DOS-Version 2.7). Über IEC-Bus lassen sich die

Floppys problemlos an Commodore 64 und VC 20 anschließen, so daß auch diese Computer 'on line' auf mehrere Megabyte zugreifen können.

Ein weiterer Vorteil der großen CBM-Floppys ist ihr doppelt so großer Pufferspeicher. Dadurch sind Sie in der Lage, mehr Dateien gleichzeitig offen zu halten als dies mit der VC 1541 möglich ist. Hier können Sie gleichzeitig bis zu 5 sequentielle Dateien oder bis zu 3 relative Dateien offen halten, natürlich auch eine Kombination daraus, z.B. 2 relative und 2 sequentielle.

Im folgenden werden die unterschiedliche Lage und Aufbau von BAM und Directory beim 1541/4040 Format mit dem 8050/ 8250-Format verglichen.
Beim 8050/8250-Format werden die Tracks 38 und 39 für BAM und Directory benutzt. In Track 39 Sektor 0 stehen der Diskettenname und das Formatkennzeichen.

```
>:00  26 00 43 00 00 00 43 42 &.C...CB
>:08  4E 20 38 30 35 30 A0 A0 M 8050
>:10  A0 A0 A0 A0 A0 A0 A0 A0
>:18  30 31 A0 32 43 A0 A0 A0 01 2C
```

In Byte 0 und 1 steht der Track/Sektor-Zeiger auf den ersten BAM Block (Track 38 Sektor 0). Byte 2 enthält das Formatkennzeichen 'C'. Byte drei bis 5 sind ungenutzt. Von Byte 6 bis 21 steht der Diskettenname, aufgefüllt mit 'Shift Space', in unserem Falle 'CBM 8050'. Byte 24 und 25 enthalten die ID '01', während in Byte 27 und 28 das DOS-Format '2C' steht. Der Rest des Blocks ist unbenutzt.

Die BAM passt hier nicht mehr in einen Block und wird daher über Track 38 verteilt; bei der 8050 werden Sektor 0 und 3 benutzt, bei der 8250 zusätzlich noch Sektor 6 und 9. Da hier auch mehr Sektoren pro Track benutzt werden, mußte der BAM-Eintrag für jede Spur vergrößert werden und belegt jetzt 5 Byte. Dabei enthält das jeweils erste Byte wieder die Anzahl der freien Sektoren pro Track und die nachfolgenden Bytes enthalten das Bitmuster der freien und belegten Sektoren (0 = Sektor belegt, 1 = Sektor frei). Hier haben wir den Inhalt von Track 38 Sektor 0.

```
>:00  26 03 43 00 01 33 1D FF
>:08  FF FF 1F 1D FF FF FF 1F
>:10  1D FF FF FF 1F 1D FF FF
>:18  FF 1F 1D FF FF FF 1F 1D
>:20  FF FF FF 1F 1D FF FF FF
>:28  1F 1D FF FF FF 1F 1D FF
>:30  FF FF 1F 1D FF FF FF 1F
>:38  1D FF FF FF 1F 1D FF FF
>:40  FF 1F 1D FF FF FF 1F 1D
>:48  FF FF FF 1F 1D FF FF FF
>:50  1F 1D FF FF FF 1F 1D FF
>:58  FF FF 1F 1D FF FF FF 1F
>:60  1D FF FF FF 1F 1D FF FF
```

```
>:68  FF 1F 1D FF FF FF 1F 1D
>:70  FF FF FF 1F 1D FF FF FF
>:78  1F 1D FF FF FF 1F 1D FF
>:80  FF FF 1F 1D FF FF FF 1F
>:88  1D FF FF FF 1F 1D FF FF
>:90  FF 1F 1D FF FF FF 1F 1D
>:98  FF FF FF 1F 1D FF FF FF
>:A0  1F 1D FF FF FF 1F 1D FF
>:A8  FF FF 1F 18 FC F3 EF 1F
>:B0  00 00 00 00 00 00 00 00
>:B8  00 00 00 00 00 00 00 0F
>:C0  F4 93 46 1A 18 6C FB FF
>:C8  1F 00 00 00 00 00 00 00
>:D0  00 00 00 00 00 00 00 00
>:D8  05 00 00 4D 04 1B FF FF
>:E0  FF 07 1B FF FF FF 07 1B
>:E8  FF FF FF 07 1B FF FF FF
>:F0  07 1B FF FF FF 07 1B FF
>:F8  FF FF 07 1B FF FF FF 07
```

Die Bytes 0 und 1 zeigen wieder auf den nächsten BAM-Block, hier Track 38 Sektor 3. Byte 2 enthält wieder das Formatkennzeichen 'C'. In Byte 4 stehen die Tracknummern, für die dieser BAM-Teil zuständig ist; hier Track 1 bis 51. Ab Position 6 finden wir die 5-Byte-Einträge für jede Spur. Der nächste BAM-Block ist analog aufgebaut, ist bei der 8050 für die Tracks 52 bis 77 zuständig und belegt die Bytes bis 140. Der letzte BAM-Block zeigt immer auf den ersten Directory-Block: Track 39, Sektor 1.

Bei der 8250 sind 4 Blocks für die BAM erforderlich, Track 38 Sektor 0 enthält die Tracks 1 bis 51, Track 38 Sektor 3 enthält 52 bis 100, Track 38 Sektor 6 enthält Track 101 bis 150 und Track 38 Sektor 9 schließlich ist für die Tracks 151 bis 154 zuständig.

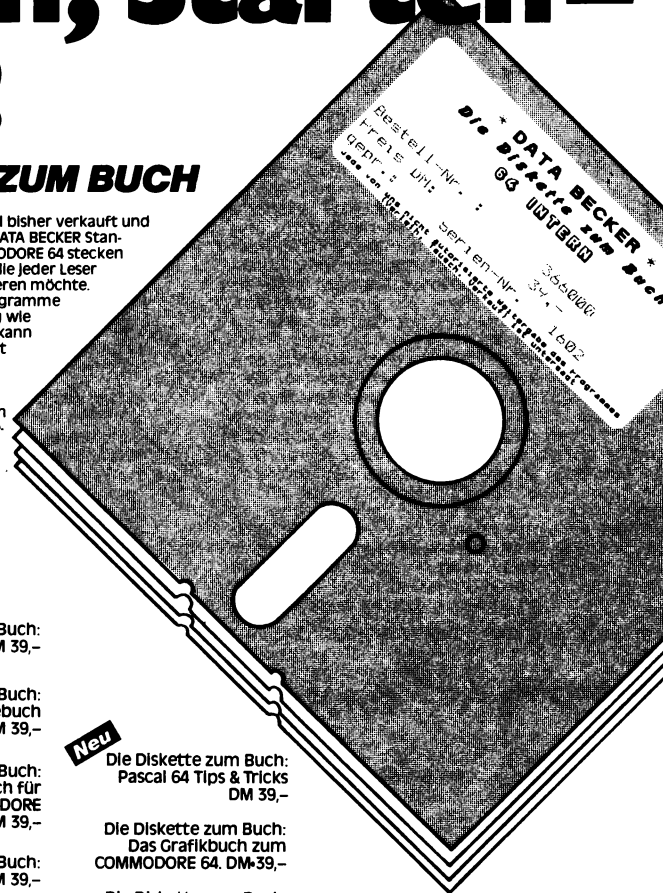
Die Directoryspur, Track 39, enthält noch 28 freie Blocks; es sind deshalb 28*8 = 224 Directoryeinträge möglich im Gegensatz zu 144 bei 1541/4040. Der Aufbau der Directory ist bei allen Formaten gleich. Im folgenden die Track-Sektor-Belegung noch einmal tabellarisch:

	1541 / 4040	8050 / 8250	
Tracks	1 - 17 : 0 - 20	1 - 39 : 0 - 28	Sektoren
	18- 24 : 0 - 18	40 - 53 : 0 - 26	
	25- 30 : 0 - 17	54 - 64 : 0 - 24	
	31- 35 : 0 - 16	65 - 77 : 0 - 22	
		nur 8250	
		78 -116 : 0 - 28	
		117 -130 : 0 - 26	
		131 -141 : 0 - 24	
		142 -154 : 0 - 22	
Blocks	683	2083 / 4186	
freie Blocks	664	2052 / 4133	

Laden, Starten – Klar!

DIE DISKETTE ZUM BUCH

Über 500.000 DATA BECKER Bücher sind bisher verkauft und das nicht ohne Grund. Die beliebten DATA BECKER Standardwerke zum VC 20 und zum COMMODORE 64 stecken voller Programmiertips und Listings, die jeder Leser am liebsten sofort am Gerät ausprobieren möchte. Doch ohne fleißiges Abtippen der Programme läuft nichts. Abtippen ist so langweilig wie unzuverlässig; der kleinste Tippfehler kann den ganzen Spaß verderben. Ab sofort nimmt Ihnen DATA BECKER diese Arbeit ab. Die DISKETTEN ZUM BUCH enthalten alle Programme und Utilities, die Sie als Listing im jeweiligen Buch finden. Diskette ins Laufwerk, gewünschtes Programm laden, starten und schon können Sie mit der ausgetüftelten Software der DATA BECKER Autoren arbeiten. Und: Sie haben die Sicherheit, daß diese Programme wirklich auf Anhieb laufen. Ist das nichts?



Die Diskette zum Buch:
Das große Drucker-Buch. DM 39,–

Die Diskette zum Buch:
Das Maschinensprachebuch zum COMMODORE 64. DM 39,–

Die Diskette zum Buch:
Das Maschinensprachebuch für Fortgeschrittene zum COMMODORE 64. DM 39,–

Die Diskette zum Buch:
Das große Floppy-Buch. DM 39,–

Die Diskette zum Buch:
64 Tips & Tricks. DM 39,–

Die Diskette zum Buch: Das Schulbuch zum COMMODORE 64. Diese Superdiskette enthält zusätzlich noch 14 weitere Lernprogramme und ein ca. 70seitiges Handbuch. Ein absolutes Muß für Schüler, Lehrer und Eltern. DM 49,–

Neu

Die Diskette zum Buch:
Pascal 64 Tips & Tricks
DM 39,–

Die Diskette zum Buch:
Das Grafikbuch zum COMMODORE 64. DM 39,–

Die Diskette zum Buch:
64 INTERN. DM 39,–

Die Diskette zum Buch:
64 für Profis. DM 39,–

Die Diskette zum Buch:
VC-20 Tips & Tricks. DM 39,–

Die Diskette zum Buch:
APPLE II Tips & Tricks. DM 39,–

Neu

Die Diskette zum Buch:
Das Trainingsbuch zu Datamat
DM 39,–

Neu

Die Diskette zum Buch:
Adventures –
und wie man sie programmiert
DM 49,–

DATA BECKER BÜCHER & PROGRAMME erhalten Sie im Computer-Fachhandel, in guten Buchhandlungen und in den Fachabteilungen der Kauf- und Warenhäuser.

DATA BECKER

Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 310010 · im Hause AUTO BECKER

DATA BECKER'S NEUE BÜCHER UND PROGRAMME FÜR COMMODORE

Spickzettel ade.

Ein neues DATA BECKER BUCH, das den Einsatz des COMMODORE 64 in der Schule entscheidend mitprägen dürfte, wurde von Professor Voß geschrieben. Besonders für Schüler der Mittel- und Oberstufe geschrieben, enthält das Buch viele interessante Problemlösungs- und Lernprogramme, die besonders ausführlich und leicht verständlich beschrieben sind. Sie ermöglichen ein intensives und anregendes Lernen, unter anderem mit folgenden Themen: Satz des Pythagoras, quadratische Gleichungen, geometrische Reihen, Pendelbewegungen, mechanische Hebel, Molekülbildung, exponentielles Wachstum, Vokabeln lernen, unregelmäßige Verben, Zinseszinsrechnung. Ein kurzer Überblick über die Grundlagen der EDV, eine knappe Wiederholung der wichtigsten BASIC-Elemente und eine Einführung in die Grundzüge der Problemanalyse vervollständigen das Ganze. Mit diesem Buch machen die Hausaufgaben wieder Spaß!

DAS SCHULBUCH ZUM COMMODORE 64, 1984, über 300 Seiten, DM 49,–

Tempo!

MASCHINENSPRACHE FÜR FORTGESCHRITTENE ist bereits das zweite Buch von Lothar Englisch zum Thema Maschinenprogrammierung mit dem COMMODORE 64. Hier wird von der Problemanalyse bis zum Maschinensprachealgorithmus in die Grundlagen der professionellen Maschinenspracheprogrammierung eingeführt. In diesem Buch finden Sie unter anderem folgende Themen behandelt: Problemlösungen in Maschinensprache, Programmierung von Interruptroutinen, Interruptquellen beim COMMODORE 64, Interrupts durch CIA's und Videocontroller, Programmierung der Ein-Ausgabe-Bausteine, die CIA's des COMMODORE 64, Timer, Echtzeituhr, parallele und serielle Ein/Ausgabe, BASIC-Erweiterungen, Programmierung eigener BASIC-Befehle und -Funktionen, Möglichkeiten zur Einbindung ins Betriebssystem sowie viele weitere Tips & Tricks zur Maschinenprogrammierung. Dieses Buch sollte jeder haben, der wirklich intensiv mit der Maschinensprache des COMMODORE 64 arbeiten will.

MASCHINENSPRACHE FÜR FORTGESCHRITTENE, 1984, ca. 200 Seiten, DM 39,–

Macht Druck.

DAS GROSSE DRUCKERBUCH für Drucker-Anwender mit COMMODORE-Computern ist endlich da! Es enthält eine riesige Sammlung von Tips & Tricks, Programmlistings und Hardwareinformationen. Rolf Brückmann und Klaus Gerits beschäftigen sich mit Sekundäradressen, Anschluß einer Schreibmaschine am Userport, Druckerschnittstellen (Centronics, V 24, IEC-Bus), hochauflösender Grafik, Text- und Grafikhardcopy, Grafik mit Standardzeichensatz, formatierter Datenausgabe, Plakatschrift, Textverarbeitung und vieles mehr. Zusätzlich wird das Betriebssystem des MPS801 zerlegt, mit Prozessorbeschreibung (8035), Blockschaltbild und einem ausführlich kommentierten ROM-Listing. Thomas Wiens schrieb den Teil über die Programmierung des Plotters VC-1520: Handhabung des Plotters, Programmierung von Sonderzeichen, Funktionendarstellung, Kuchen und Säulendiagramme, Entwurf dreidimensionaler Gegenstände. Natürlich wieder viele interessante Listings. Unentbehrlich für jeden, der einen COMMODORE 64 oder VC-20 und einen Drucker besitzt.

DAS GROSSE DRUCKERBUCH, 1984, über 300 Seiten, DM 49,–

Tausendsassa.

Fast alles, was man mit dem COMMODORE 64 machen kann, ist in diesem Buch ausführlich beschrieben. Es ist nicht nur spannend zu lesen wie ein Roman, sondern enthält neben nützlichen Programmlistings vor allem viele, viele Anwendungsmöglichkeiten des C64. Dabei wurde besonderer Wert darauf gelegt, daß das Buch auch für Laien leicht verständlich ist. Eine Auswahl aus der Themenvielfalt: Gedichte vom Computer, Einladung zur Party, Diplomarbeit – professionell gestaltet, individuelle Werbebriefe, Autokosten im Griff, Baukostenberechnung, Taschenrechner, Rezeptkartei, Lagerliste, persönliches Gesundheitsarchiv, Diätplan elektronisch, intelligentes Wörterbuch, kleine Notenschule, CAD für Handarbeit, Routenoptimierung, Schaufensterwerbung, Strategiespiele. Teilweise sind Programmlistings fertig zum Eintippen enthalten, soweit sich die „Rezepte" auf 1–2 Seiten realisieren ließen. Wenn Sie bisher nicht immer wußten, was Sie mit Ihrem 64er alles anfangen sollten, nach dem Lesen des IDEENBUCHES wissen Sie's bestimmt!

DAS IDEENBUCH ZUM COMMODORE 64, 1984, über 200 Seiten, DM 29,–

Prof. 64.

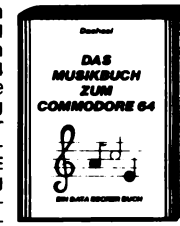
Ein faszinierendes Buch, um in die Welt der Wissenschaft einzusteigen, hat Rainer Severin geschrieben. Zunächst werden Variablentypen, Rechengenauigkeit und nützliche POKE-Adressen des COMMODORE 64 bezüglich den Anforderungen wissenschaftlicher Probleme analysiert. Verschiedene Sortieralgorithmen wie Bubble, Quick und Shell-Sort werden miteinander verglichen. Die Programmbeispiele aus der Mathematik nehmen dabei eine zentrale Stelle im Buch ein: Nullstellen nach Newton, numerische Ableitung mit dem Differenzenquotienten, lineare und nichtlineare Regression, Chi-Quadrat-Verteilung und Anpassungstest, Fourieranalyse und -synthese, Skalar-, Vektor- und Spatprodukt, ein Programmpaket zur Matrizenrechnung für Inversion, Eigenwerte und vieles weitere mehr. Programme aus der Chemie (Periodensystem), Physik, Biologie (Schadstoffe in Gewässern – Erfassung der Meßwerte), Astronomie (Planetenpositionen) und Technik (Berechnung komplexer Netzwerke, Platinenlayout am Bildschirm) und viele weitere Softwarelistings zeigen die riesigen Möglichkeiten auf, die der Computer in Wissenschaft und Technik hat.

COMMODORE 64 FÜR TECHNIK UND WISSENSCHAFT, 1984, über 200 Seiten, DM 49,–

Grundkurs.

Das neue BASIC-Trainingsbuch zum C-64 ist eine ausführliche, didaktisch gut geschriebene Einführung in das CBM BASIC V2. Alle Befehle werden ausführlich erläutert. Dieses Buch geht aber über eine reine Befehlsbeschreibung hinaus, es wird eine fundierte Einführung in die Programmierung gegeben. Von der Problemanalyse bis zum fertigen Algorithmus lernt man das Entwerfen eines Programmes und den Entwurf von Datenflußplänen. ASCII-Code und verschiedene Zahlensysteme wie hexadezimal, binär und dezimal sind nach der Lektüre des Buches keine Fremdworte mehr. Die Programmierung von Schleifen, Sprüngen, bedingten Sprüngen lernt man leicht durch „learning by doing". So enthält das Trainingsbuch viele Aufgaben, Übungen und unzählige Beispiele. Den Schluß des Buches bildet eine Einführung ins professionelle Programmieren, in der es um mehrdimensionale Felder, Menuesteuerung und Unterprogrammtechnik geht. Endlich ein Buch, das Ihnen wirklich hilft, solide und sicher BASIC zu lernen.

BASIC TRAININGSBUCH ZUM COMMODORE 64, 1984, ca. 250 Seiten, DM 39,–

Sang und Klang!

Der COMMODORE 64 ist ein Musikgenie. DAS MUSIKBUCH hilft Ihnen, die riesigen Klangmöglichkeiten des C64 zu nutzen. Die Themenbreite reicht von einer Einführung in die Computermusik über die Erklärung der Hardwaregrundlagen des COMMODORE 64 und die Programmierung in BASIC bis hin zur fortgeschrittenen Musikprogrammierung in Maschinensprache. Einiges aus dem Inhalt: Soundregister des COMMODORE 64, Gate-Signal, Programmierung der "ADSR"-Werte, Synchronisation und Ring-Modulation, Counterprinzip, lineare und nichtlineare Musikprogrammierung, Frequenzmodulation, Interrupts in der Musikprogrammierung und vieles mehr. Zahlreiche Beispielprogramme, komplette Songs und nützliche Routinen ergänzen den Text. Geschrieben wurde das Buch von Thomas Dachsel, dem Autor der weltbekannten Musikprogramme Synthimat und Synthesound. Erschließen Sie sich die Welt des Sounds und der Computermusik mit dem Musikbuch zum C-64!

DAS MUSIKBUCH ZUM COMMODORE 64, über 200 Seiten, DM 39,–

Nützlich.

Das Trainingsbuch zu MULTIPLAN bietet eine gute Einführung in die Grundlagen der Tabellenkalkulation. Dabei wird großer Wert auf ein möglichst schnelles Einarbeiten in die wichtigsten Befehle gelegt, so daß man bald sicher mit MULTIPLAN arbeiten kann, ob nun auf dem COMMODORE 64 oder einem anderen Rechner. Am Ende wird man in der Lage sein, den umfangreichen Befehlssatz von MULTIPLAN auch kommerziell zu nutzen. Übungen am Ende jedes Kapitels sorgen dafür, daß man das Gelernte lange behält. Grundlage des Buches sind viele Seminare, die der Autor zu MULTIPLAN konzipiert und erfolgreich durchgeführt hat.

DAS TRAININGSBUCH ZU MULTIPLAN, 1984, ca. 250 Seiten, DM 49,–

Für Tüftler.

Ein hochinteressantes Buch für Hobbyelektroniker hat Rolf Brückmann vorgelegt. Er ist ein engagierter Techniker, für den der Computer Hobby und Beruf zur gleichen Zeit ist. Vor allem aber kennt er den C-64 in- und auswendig. So werden einführend die Schnittstellen des COMMODORE 64 detailliert beschrieben und kurz die Funktionsweise der CIAs 6526 erläutert. Hauptteil des Buches sind die Beschreibungen der vielfältigen Einsatzmöglichkeiten des COMMODORE 64. Die vielen Schaltungen, von Rolf Brückmann alle selbst

entwickelt, sind jeweils umfangreich dokumentiert und leichtverständlich erklärt. Die Reihe der hier ausführlich behandelten Anwendungen mit dem COMMODORE 64 ist äußerst umfangreich: Motorsteuerung, Stoppuhr mit Lichtschranke, Lichtorgel, A/D-Wandler, Spannungsmessung, Temperaturmessung und vieles mehr. Dazu kommen noch eine Reihe kompletter Schaltungen zum Selberbauen, wie ein EPROM Programmiergerät für den C-64, eine EPROM-Karte, ein Frequenzzähler und Sprachein/ausgabe (!). Zusätzlich sind jeweils Schaltplan, Softwarelisting und zu einigen Schaltungen sogar zusätzlich Platinenlayouts vorhanden.

DER COMMODORE 64 UND DER REST DER WELT, 1984, ca. 220 Seiten, DM 49,–

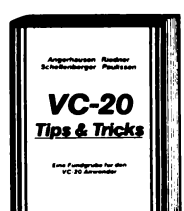
Computerkünstler.

Das Grafikbuch zum COMMODORE 64 Buch aus der Bestseller-Serie von DATA BECKER stammt aus der Feder von Axel Plenge. Es geht weit über die reine Hardware-Beschreibung der Grafikeigenschaften des C-64 hinaus. Der Inhalt reicht von den Grundlagen der Grafikprogrammierung bis zum Computer Aided Design. Es ist ein Buch für alle, die mit ihrem C-64 kreativ tätig sein wollen. Themen sind z. B.: Zeichensatzprogrammierung, bewegte Sprites, High-Resolution, Multicolor-Grafik, Lightpenanwendungen, Betriebsarten des VIC, Verschieben der Bildschirmspeicher, IRQ-Handhabung, 3-Dimensionale Grafik, Projektionen, Kurven-, Balken- und Kuchendiagramme, Laufschriften, Animation, bewegte Bilder. Viele Programmlistings und Beispiele sind selbstverständlich. Das COMMODORE-BASIC V2 unterstützt die herausragenden Grafikeigenschaften des C-64 bekanntlich kaum. Hier helfen die vielen Beispielprogramme in diesem Buch weiter, die die faszinierende Welt der Computergrafik jedermann zugänglich machen. Kompetent ist der Autor dazu wie kaum ein anderer, schließlich hat er das äußerst leistungsfähige Programm SUPERGRAFIK geschrieben.

DAS GRAFIKBUCH ZUM COMMODORE 64, 1984, 295 Seiten, DM 39,–

Vielfalt.

Auf dem neuesten Stand ist VC-20 TIPS & TRICKS von Dirk Paulissen gebracht worden, der über hundert Seiten hinzufügte. Bisher schon enthalten waren Informationen über Speicheraufbau des VC-20 und die Erweiterungsmöglichkeiten, ein Grafikkapitel über programmierbare Zeichen, Laufschrift und die Supererweiterung. Stark erweitert wurde der Abschnitt über POKEs und andere nützliche Routinen. Ob es um die Programmierung der Funktionstasten, Programme die sich selber starten, „Maus"-Simulation mit dem Joystick oder die Änderung von Speicherbereichen geht, man ist immer wieder über die Fülle der Möglichkeiten erstaunt. Der Clou dieses Buches sind aber die vielen Programmlistings. Die BASIC-Erweiterungen allein stellen schon ein erstklassiges Toolkit dar: APPEND (Anhängen von Programmen, AUTO (automatische Zeilennummerierung), BASIC-Befehle auf Tastendruck, PRINT POSITION, UNNEW, Strings größer als 88 Zeichen einlesen und vieles mehr. Die Bandbreite reicht von Spielen wie Goldgräber oder Starshooter bis zu nützlichen Programmen wie Cassetteninhaltsverzeichnis und -katalog mit automatischem Suchen nach Dateien und einem Terminkalender. Für den VC-20 Anwender ist dieser 324 Seiten-Wälzer eine wahre Fundgrube, in der es immer etwas neues zu entdecken gibt.



VC-20 TIPS & TRICKS, 3. erweiterte und überarbeitete Auflage, 1984, 324 Seiten, DM 49,–

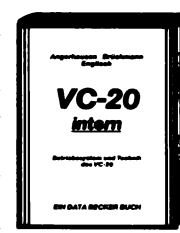
Interessant.

Einen guten Einstieg in PASCAL bietet dieses Trainingsbuch. Es gibt eine leichtverständliche Einführung, sowohl in UCSD-PASCAL wie auch in PASCAL64, wobei allerdings EDV-und BASIC-Grundkenntnisse vorausgesetzt werden. Der Autor, Ottmar Korbmacher, ist Student der Mathematik. Ihm gelingt es, in einem sprachlich aufgelockerten Stil mit vielen interessanten Beispielprogrammen, dem Leser Programmstrukturen, Ein/Ausgabe, Arithmetik und Funktionen, Prozeduren und Rekursionen, Sets, Files und Records näherzubringen. Die Übungsaufgaben am Ende jeden Kapitels helfen dabei, das Gelernte zu vertiefen. Ein Anhang mit allen PASCAL-Schlüsselworten, der ansich schon ein umfangreiches Lexikon darstellt, macht das Buch für jeden PASCAL-Anwender interessant.

DAS TRAININGSBUCH ZU PASCAL, 1984, ca. 250 Seiten, DM 39,–

Bewährt.

Die bereits dritte Auflage von VC-20 INTERN ist wieder erheblich erweitert worden. Das Buch beschäftigt sich ausführlich mit der Technik und dem Betriebssystem des VC-20. Dazu gehört natürlich zuerst einmal ein ausführlich dokumentiertes ROM-Listing. Dazu gehört auch die Belegung der Zeropage, dem wichtigsten Speicherbereich für den 6502-Prozessor, eine übersichtliche Auflistung der Adressen aller Betriebssystemroutinen, ihrer Bedeutung und ihrer Übergabeparameter. Dies ermöglicht dem Programmierer endlich, den VC-20 von Maschinensprache aus sinnvoll einzusetzen. Denn warum Routinen, die bereits vorhanden sind, noch einmal schreiben? Weiterer Inhalt: Einführung in die Maschinensprache – Maschinensprachemonitor, Assembler, Disassembler – Verbindung von Maschinensprache- und BASIC-Programmen – Beschreibung der wichtigen IC's des VC-20 – Blockschaltbild – drei Original COMMODORE-Schaltpläne. Das Buch braucht jeder der sich intensiv mit der Maschinenspracheprogrammierung des VC-20 auseinandersetzen möchte.



VC-20 INTERN, 3. Auflage, 1984, ca. 230 Seiten, DM 49,–

Starthilfe!

Das sollte Ihr erstes Buch zum COMMODORE 64 sein: 64 FÜR EINSTEIGER ist eine sehr leicht verständliche Einführung in Handhabung, Einsatz, Ausbaumöglichkeiten und Programmierung des COMMODORE 64, die keinerlei Vorkenntnisse voraussetzt. Sie reicht vom Anschluß des Geräts über die Erklärung der einzelnen Tasten und Funktionen sowie die Peripheriegeräte und ihre Bedienung bis zum ersten Befehl. Schritt für Schritt führt das Buch Sie in die Programmiersprache BASIC ein, wobei Sie nach und nach eine komplette Adressenverwaltung erstellen, die Sie anschließend nutzen können. Zahlreiche Abbildungen und Bildschirmfotos ergänzen den Text. Viele Anwendungsbeispiele geben nützliche Anregungen zum sinnvollen Einsatz des COMMODORE 64. Das Buch ist sowohl als Einführung als auch als Orientierung vor dem 64er Kauf gut geeignet.

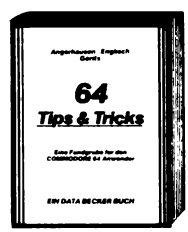
64 FÜR EINSTEIGER, 1984, ca. 200 Seiten, DM 29,-

Von A bis Z.

So etwas haben Sie gesucht: Umfassendes Nachschlagewerk zum COMMODORE 64 und seiner Programmierung. Allgemeines Computerlexikon mit Fachwissen von A-Z und Fachwörterbuch mit Übersetzungen wichtiger englischer Fachbegriffe - das DATA BECKER LEXIKON ZUM COMMODORE 64 stellt praktisch drei Bücher in einem dar. Es enthält eine unglaubliche Vielfalt an Informationen und dient so zugleich als kompetentes Nachschlagewerk und als unentbehrliches Arbeitsmittel. Viele Abbildungen und Beispiele ergänzen den Text. Ein Muß für jeden COMMODORE 64 Anwender!

DAS DATA BECKER LEXIKON ZUM COMMODORE 64, 1984, 354 Seiten, DM 49,-

Fundgrube.



64 Tips & Tricks ist eine hochinteressante Sammlung von Anregungen zur fortgeschrittenen Programmierung des COMMODORE 64, POKE's und andere nützliche Routinen, interessanten Programmen sowie interessanten Programmiertips & -tricks. Aus dem Inhalt: 3D-Graphik in BASIC - Farbige Balkengraphik - Definition eines eigenen Zeichensatzes - Tastaturbelegung und ihre Änderung - Dateneingabe mit Komfort - Simulation der Maus mit einem Joystick - BASIC für Fortgeschrittene - C-64 spricht deutsch - CP/M auf dem COMMODORE 64 - Druckeranschluß über den USER-Port - Datenübertragung von und zu anderen Rechnern - Expansion-Port - Synthesizer in Stereo - Retten einer nicht ordnungsgemäß geschlossenen Datei - Erzeugen einer BASIC-Zeile in BASIC - Kassettenpuffer als Datenspeicher - Sortieren von Stringfelder - Multitasking auf dem COMMODORE 64 - POKE's und die Zeropage - GOTO, GOSUB und RESTORE mit berechneten Zeilennummern, INSTR und STRING-Funktion - Repeat-Funktion für alle Tasten - und vieles andere mehr. Alle Maschinenprogramme mit BASIC-Ladeprogrammen. 64 Tips & Tricks ist eine echte Fundgrube für jeden COMMODORE 64 Anwender. Schon über 65000mal verkauft!

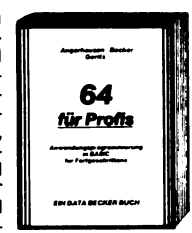
64 TIPS & TRICKS, 1984, über 300 Seiten, DM 49,-

Know-how!

350 Seiten dick ist die 4. erweiterte und überarbeitete Auflage von 64 INTERN geworden. Das bereits über 65000mal verkaufte Standardwerk bietet jetzt noch mehr Informationen. Hinzugekommen ist ein Kapitel über den IEC-Bus und viele, viele Ergänzungen, die sich im Laufe der Zeit angesammelt haben. Ebenfalls überarbeitet und noch ausführlicher ist jetzt die Dokumentation des ROM-Listings. Weitere Themen: genaue Beschreibung des Sound- und Video-Controllers mit vielen Hinweisen zur Programmierung von Sound und Grafik, der Ein/Ausgabesteuerung (CIAs), BASIC-Erweiterungen (RENEW, HARDCOPY, PRINTUSING), Hinweise zur Maschinenprogrammierung wie Nutzung der E/A-Routinen des Betriebssystems, Programmierung der Schnittstelle RS 232, ein Vergleich VC20 - C-64 - CBM zur Umsetzung von Programmen. Dies und viele weitere Informationen machen das umfangreiche Werk zu einem unentbehrlichen Arbeitsmittel für jeden, der sich ernsthaft mit Betriebssystem und Technik des C-64 auseinandersetzen will. Zum professionellen Gehalt des Buches tragen auch zwei Original-COMMODORE-Schaltpläne zum Ausklappen und zahlreiche ausführlich beschriebene und dokumentierte Fotos, Schaltbilder und Blockdiagramme bei.

64 INTERN, 4. überarbeitete und erweiterte Auflage, 1984, ca. 350 Seiten, DM 69,-

Erfolgreich.



64 für Profis zeigt, wie man erfolgreich Anwendungsprobleme in BASIC löst und verrät die Erfolgsgeheimnisse der Programmierprofis. Vom Programmentwurf über Menüsteuerung, Maskenaufbau, Parametrisierung, Datenzugriff und Druckausgabe bis hin zur guten Dokumentation wird anschaulich mit vielen Beispielen dargestellt wie Profi-Programmierung vor sich geht. Besonders stolz sind wir auf die völlig neuartige Datenzugriffsmethode QUISAM, die in diesem Buch zum ersten Mal vorgestellt wird. QUISAM erlaubt eine beliebige Datensatzlänge, die dynamisch mit der Eingabe der Daten wächst. Eine lauffertige Literaturstellenverwaltung veranschaulicht die Arbeitsweise von QUISAM. Neben diesem Programm finden Sie noch weitere Programme zur Lager- und Adressenverwaltung, Textverarbeitung und einen Reportgenerator. Alle diese Programme sind mit Variablenliste versehen und ausführlich beschrieben. Damit sind diese für Ihre Erweiterungen offen und können von Ihnen an Ihre persönlichen Bedürfnisse angepaßt werden. Steigen Sie in die Welt der Programmierprofis ein.

64 FÜR PROFIS, 2. Auflage, 1984, ca. 300 Seiten, DM 49,-

Rundum gut!

Endlich ein Buch, das Ihnen ausführlich und verständlich die Arbeit mit der Floppy VC-1541 erklärt. Das große Floppybuch ist für Anfänger, Fortgeschrittene und Profis gleichermaßen. Interessant. Sein Inhalt reicht von der Programmspeicherung bis zum DOS-Zugriff, von der sequentiellen Datenspeicherung bis zum Direktzugriff, von der technischen Beschreibung bis zum ausführlich dokumentierten DOS-Listing, von den Systembefehlen bis zur detaillierten Beschreibung der Programme auf der Test-Demo-Diskette. Exakt beschriebene Beispiel- und Hilfsprogramme ergänzen dieses neue Superbuch. Aus dem Inhalt: Speichern von Programmen – Floppy-Systembefehle – Sequentielle Datenspeicherung – relative Datenspeicherung – Fehlermeldungen und Ihre Ursachen – Direktzugriff – DOS-Listing der VC-1541 – BASIC-Erweiterungen und Programme – Overlaytechnik – Diskmonitor – IEC-Bus und serieller Bus – Vergleich mit den großen CBM-Floppies. Ein Muß für jeden Floppy-Anwender! Bereits über 45.000mal verkauft.

DAS GROSSE FLOPPY-BUCH, 2. überarbeitete Auflage, 1984, ca. 320 Seiten, DM 49,–

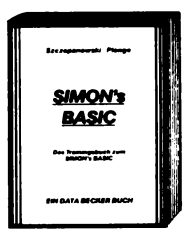
Füttern erwünscht!

Diese beliebte umfangreiche Programmsammlung hat es in sich. Über 50 Spitzenprogramme für den COMMODORE 64 aus den unterschiedlichsten Bereichen, von attraktiven Superspielen (Senso, Pengo, Master Mind, Seeschlacht, Poisson Square, Memory) über Grafik- und Soundprogramme (Fourier 64, Akustograph, Funktionsplotter) und mathematische Programme (Kurvendiskussion, Dreieck) sowie Utilities (SORT, RENUMBER, DISK INIT, MENUE) bis hin zu kompletten Anwendungsprogrammen wie „Videothek", „File Manager" und einer komfortablen Haushaltsbuchführung, in der fast professionell gebucht wird. Der Hit zu jedem Programm sind aktuelle Programmiertips und Tricks der einzelnen Autoren zum Selbermachen. Also nicht nur abtippen, sondern auch dabei lernen und wichtige Anregungen für die eigene Programmierung sammeln.

DATA BECKER's GROSSE 64er PROGRAMMSAMMLUNG, 1984, 250 Seiten, DM 49,–

Bestseller aus bester Hand

BASIC-PLUS.



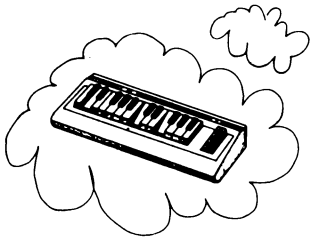
SIMON's BASIC ist ein Hit – wenn man es richtig nutzen kann. Auf über 300 Seiten erklärt Ihnen das DATA BECKER Trainingsbuch detailliert den Umgang mit den über 100 Befehlen des SIMON's BASIC. Alle Befehle werden ausführlich dargestellt, auch die, die nicht im Handbuch stehen! Natürlich zeigen wir auch die Macken des SIMON's BASIC und geben wichtige Hinweise wie man diese umgeht. Natürlich enthält das Buch viele Beispielprogramme und viele interessante Programmiertricks. Weiterer Inhalt: Einführung in das CBM-BASIC 2.0 – Programmierhilfen – Fehlerbehandlung – Programmschutz – Programmstruktur – Variablen – Zahlenbehandlung – Eingabekontrolle – Ein/Ausgabe Peripheriebefehle – Graphik – Zeichensatzerstellung – Sprites – Musik – SIMON's BASIC und die Verträglichkeit mit anderen Erweiterungen und Programmen. Dazu ein umfangreicher Anhang. Nach jedem Kapitel finden Sie Testaufgaben zum optimalen Selbststudium und zur Lernerfolgskontrolle.

DAS TRAININGSBUCH ZUM SIMON's BASIC, 2. überarbeitete Auflage, 1984, ca. 380 Seiten, DM 49,–

Schrittmacher.

Eine leicht verständliche Einführung in die Maschinenspracheprogrammierung für alle, denen das C-64 BASIC nicht mehr ausreicht. Sie lernen Aufbau und Arbeitsweise des 6510-Mikroprozessors kennen und anwenden. Dabei werden die Analogien zu BASIC Ihnen beim Verständnis helfen. Ein weiteres Kapitel beschäftigt sich mit der Eingabe von Maschinenprogrammen. Dort erfahren Sie auch alles über Monitor-Programme sowie über Assembler. Zum einfachen und komfortablen Erstellen Ihrer eigenen Maschinensprache enthält das Buch einen kompletten ASSEMBLER, damit Sie gleich von Anfang an komfortabel und effektiv programmieren können. Weiterhin finden Sie dort einen DISASSEMBLER, mit dem Sie sich Ihre Maschinenprogramme oder die Routinen des BASIC-Interpreters und des BASIC-Betriebssystems ansehen können. Ein besonderer Clou ist ein in BASIC geschriebener Einzelschrittsimulator, mit dem Sie Ihre Programme schrittweise ausführen können. Dabei werden Sie nach jedem Schritt über Registerinhalte und Flags informiert und können den logischen Ablauf Ihres Programmes verfolgen. Eine unschätzbare Hilfe, besonders für den Anfänger. Als Beispielprogramm finden Sie ausführlich beschriebene Routinen zur Grafikprogrammierung und für BASIC-Erweiterungen. Natürlich sind alle Beispiele und Programme auf den C-64 zugeschnitten.

DAS MASCHINENSPRACHEBUCH ZUM COMMODORE 64, ca. 200 Seiten, DM 39,–



SYNTHIMAT

SYNTHIMAT verwandelt Ihren COMMODORE 64 in einen professionellen, polyphonen, dreistimmigen Synthesizer, der in seinen unglaublich vielen Möglichkeiten großen Systemen kaum nachsteht.

SYNTHIMAT in Stichworten:
drei Oszillatoren (VCOs) mit 7 Fußlagen und 8 Wellenformen – drei Hüllkurvengeneratoren (ADSRs) – ein Filter (VCF) mit 8 Betriebsarten und Resonanzregulierung – VCF mit Eingang für externe Signalquelle – ein Verstärker (VCA) – Ringmodulation mit allen drei VCOs – 8 softwaremäßig realisierte Oszillatoren (LFOs) – kräftiger Klang durch polyphones Spielen – zwei Manuale (Solo und Begleitung) – speichern von bis zu 256 Klangregistern – schneller Registerwechsel – speichern von 9 Registerdateien auf Diskette – „Bandaufnahme" auf Diskette durch direktes Spielen – keine lästige Noteneingabe – speichern von bis zu 9 „Bandaufnahmen" je Diskette – integrierte 24 Stunden-Echtzeituhr – einstellbares PITCH-BENDING – farblich gekennzeichnete, übersichtlich angeordnete Module – umfangreiches Handbuch – läuft mit einem Diskettenlaufwerk – Diskettenprogramm.

DM 99,–

STRUKTO 64

STRUKTO 64 ist eine fantastische neue Programmiersprache für strukturiertes Programmieren mit dem C-64 und für alle Programmierer geeignet, die den C-64 als Allround-Computer einsetzen und auf einfache Weise anspruchsvolle Programme erstellen wollen.

STRUKTO 64 in Stichworten:
Interpretersprache, die die Vorzüge von BASIC und PASCAL vereint – strukturiertes Programmieren – übersichtliche Programme – leichte Erlernbarkeit – einfache Bedienung – eingebautes Toolkit erleichtert das Eingeben und Verbessern von Programmen – leichteres Arbeiten mit der Floppy – Sprite-Editor ermöglicht das Einlesen der Sprite-Formen direkt vom Bildschirm – Graphikbedienung wird mit gut durchdachten Befehlen unterstützt – Abspielen von Musik ist unabhängig vom Programmablauf möglich – ca. 80 neue Befehle – lieferbar als Diskettenprogramm – ausführliches deutsches Handbuch.

DM 99,–

Für viele ein Traum, für die meisten bisher zu teuer: die Rede ist von einer echten Datenbank für den 64er. SUPERBASE 64 füllt eine Lücke. Nicht allein die Kapazität, die verwaltet werden kann, bewegt sich in professionellen Regionen, die ausgeprägten Fähigkeiten des SUPERBASE 64 im Rechnen und Kalkulieren lassen dieses Paket beinahe als Rund-Um-Software erscheinen.

SUPERBASE 64 in Stichworten:
maximale Datensatzlänge 1108 Zeichen, verteilt auf bis zu 4 Bildschirmseiten – bis zu 127 Felder pro Datensatz, wobei Textfelder bis zu 255 Zeichen lang sein können – insgesamt 15 Einzeldateien können zu einer SUPERBASE-Datenbank verknüpft werden – Speicherkapazität nur durch Diskette begrenzt – umfangreiche Auswertungsmöglichkeiten und komfortabler Report-Generator – Kalkulationsmöglichkeiten und Rechnen – Import- (Einlesen von externen Daten) und Export- (Ausgabe von SUPERBASE Dateien als sequentielle Datei) Funktionen ermöglichen Datenaustausch mit anderen Programmen – durch leistungsfähige, eigene Datenbanksprache auch als kompletter Anwendungsgenerator verwendbar.

DM 398,–

MASTER 64

MASTER 64 ist ein professionelles Programmentwicklungssystem für den C-64, das es Ihnen ermöglicht, die Programmentwicklungszeit auf einen Bruchteil der sonst üblichen Zeit zu reduzieren. MASTER 64 bietet einen Programmkomfort, den Sie nutzen sollten.

MASTER 64 in Stichworten:
70 zusätzliche Befehle – Bildschirmmaskengenerator – definieren von Bildschirmzonen – Eingabe aus Zonen – formatierte Ausgabe – Abspeicherung von Bildschirminhalten – Arbeiten mit mehreren Bildschirmmasken – ISAM Dateiverwaltung, in der Datensätze über einen Zugriffschlüssel angesprochen werden können – Datensätze bis zu 254 Zeichen – Schlüssellänge bis zu 30 Zeichen – Dateigröße nur von Diskettenkapazität abhängig – Zugriff über Schlüssel und Auswahlmasken – Bildschirm- und Druckmaskengenerator – Erstellung beliebiger Formulare und Ausgabemasken – BASIC-Erweiterungen – Toolkitfunktionen – Mehrfachgenaue Arithmetik (Rechnen mit 22 Stellen Genauigkeit).

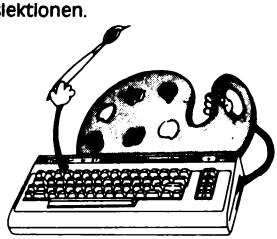
DM 198,–

TEXTOMAT

Das Bearbeiten von Texten gehört zum wichtigsten Betätigungsfeld von Homecomputer-Anwendern. So ist es auch nicht verwunderlich, daß eine Unzahl verschiedener Textprogramme für den 64er angeboten wird. TEXTOMAT zeichnet sich dadurch aus, daß er auch vom Einsteiger sofort benutzt werden kann. Über eine Menuezeile können alle Funktionen angewählt werden. Selbstverständlich beherrscht TEXTOMAT deutsche Umlaute und Sonderzeichen.

TEXTOMAT in Stichworten:
Diskettenprogramm – durchgehend menuegesteuert – deutscher Zeichensatz auch auf COMMODORE-Druckern Rechenfunktionen für alle Grundrechenarten – 24.000 Zeichen pro Text im Speicher – beliebig lange Texte durch Verknüpfung – horizontales Scrolling für 80 Zeichen pro Zeile – läuft mit 1 oder 2 Floppies – frei programmierbare Steuerzeichen – Formularsteuerung für Randeinstellung u.s.w. – komplette Bausteinverarbeitung – Blockoperationen, Suchen und Ersetzen – Serienbriefschreibung mit DATAMAT – formatierte Ausgabe auf Bildschirm – an fast jeden Drucker anpaßbar – ausführliches deutsches Handbuch mit Übungslektionen.

DM 99,–



PAINT PIC

Malen (!) mit dem Computer, welch eine faszinierende Idee. Mit dem Malprogramm PAINT PIC für den COMMODORE 64 wird diese Idee Realität. Mit PAINT PIC ist es auch für den Einsteiger leicht, fantastische Computerbilder zu erstellen. Man kann die Bilder auf Diskette abspeichern und wieder laden und selbstverständlich steht auch weiterhin der COMMODORE-Zeichensatz zur Verfügung. Wichtig: PAINT PIC benötigt keine zusätzliche Hardware.

PAINT PIC in Stichworten:
Programmsteuerung: Tastatur – Steuerung des Stifts: Cursortasten und eckige Klammer (diag.) (Joystick kann benutzt werden) – Routinen: Linien, Rechtecke, Dreiecke, Parallelogramme, Kreise, Kreisbögen, Ellipsen, Bestimmung von Mittelpunkt, und perspektivischer Linie, Kopieren und Drehen von Teilbildern, Verdoppeln, halbieren und spiegeln von Teilbildern – Modi: Malstiftmodus (schmale Linie) Pinselmodus (8 verschiedene Breiten) (Art der Linie selbst definierbar) – Textmodus (kompl. Zeichensatz COMMODORE) (Hoch-Tiefschrift) – Speichern: Teilbilder (Blöcke) oder ganze Bilder – Menue: 1 Hauptmenue mit 8 Untermenues – mit ausführlichem deutschen Handbuch – Diskettenprogramm – Bilder kann man auf Diskette abspeichern.

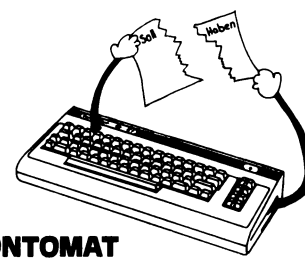
DM 99,–

PROFIMAT

Wer sich tiefer in die Innereien des Computers begeben will, kommt ohne besonderes Werkzeug nicht aus. Einerseits muß der volle Einblick in alle Speicherbereiche möglich sein, andererseits soll der Umgang mit Maschinenprogrammen so komfortabel wie möglich gestaltet sein. PROFIMAT hat Lösungen für beide Probleme: Der Maschinensprache-Monitor PROFI-MON bietet alle Hilfsmittel zum Umgang mit Maschinenprogrammen; PROFI-ASS ist ein Macro-Assembler, der das Schreiben von Maschinenprogrammen fast so einfach macht wie das Programmieren in BASIC.

PROFIMAT in Stichworten:
Registerinhalte und Flags anzeigen – Speicherinhalte anzeigen – Maschinenprogramme laden, ausführen und speichern – Speicherbereiche durchsuchen, vergleichen, füllen und verschieben – echter Einzelschrittmodus – Setzen von Unterbrechungspunkten – schneller Trace-Modus – Rückkehr zu BASIC – formatfreie Eingabe – Verkettung beliebig vieler Quellprogramme – erzeugter Objektcode kann in Speicher oder auf Diskette gehen – formatiertes Assemblerlisting – ladbare Symboltabellen – redefinierbare Symbole – Operatoren – Unterstützung der Fließkommaarithmetik – bedingte Assemblierung – Assemblerschleifen – MACROS mit beliebigen Parametern.

DM 99,–

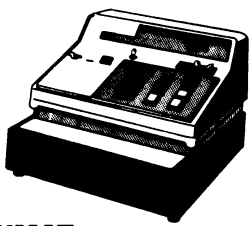


KONTOMAT

KONTOMAT ist ein menuegesteuertes Einnahme-Überschußprogramm nach §4(3) EStG mit Kassenbuch, Bankkontenüberwachung, automatischer Steuerbuchung, AFA Tabellenerstellung, Kontenblättern, Ermittlung der USt.-Voranmeldungswerte und Monats- und Jahresabrechnung. Der neue KONTOMAT ist voll parameterisiert und läßt sich damit an Ihre Bedürfnisse anpassen. Für alle Gewerbetreibenden, die nicht laut HGB zur Buchführung verpflichtet sind. KONTOMAT ist für den gewerblichen Einsatz, aber auch als Lernprogramm oder zur Haushaltsbuchführung geeignet.

KONTOMAT in Stichworten:
Diskettenprogramm – maximal 120 Konten – Beträge mit bis zu 6 Vor- und 2 Nachkommastellen – 4 Mehrwert- und Vorsteuersätze – intervallmäßige Belegeingabe – 4 Buchungsarten (SOLL, HABEN, SOLL/HABEN und HABEN/SOLL) – Anzeige der Soll- und Habensumme bei mehrfachen Buchungssätzen – komfortable Belegeingabe mit Datum, Buchungstext, Stuerkennzeichen und Betrag – Druck des Journals während der Belegeingabe – Druck von umfangreichen Kontenblättern – Druck einer Summen- und Saldenliste mit Monats- und Jahresumsatzsummen – betriebswirtschaftliche Auswertung mit Druckausgabe – Ermittlung und Druckausgabe der Umsatzsteuerzahllast – Speicherung der Anlagegüter und automatische Abschreibung am Jahresende – übersichtliche AfA-Liste – arbeitet mit 1 oder 2 Laufwerken – umfangreiches deutsches Handbuch.

DM 148,–



FAKTUMAT

Mit FAKTUMAT ist das Schreiben von Rechnungen kein Alptraum mehr. Eine Sofortfakturierung mit integrierter Lagerbuchführung. Individuelle Anpassung von Steuersätzen, Maßeinheiten und Firmendaten. Kunden- und Artikelstamm voll pflegbar. Schneller Zugriff auf Kunden- und Artikeldaten, über freidefinierbaren, 6-stelligen Schlüssel. Automatische Fortschreibung von Artikel- und Kundendaten, individuell nutzbar. Alles in allem die Arbeits- und Zeitersparnis, die Sie sich schon immer gewünscht haben.

FAKTUMAT in Stichworten:
voll menuegesteuert – läuft mit einer oder zwei Floppies – Diskettenwechsel (eine Floppy) nur beim Wechsel vom Hauptmenue ins Unterprogramm und umgekehrt – mit Ausnahme des Ausschaltens der Floppy während der Verarbeitung werden alle Fehler abgefangen (z. B. Drucker nicht eingeschaltet – arbeitet mit 1525, 1526 (?), MPS 801, EPSON Drucker und DATA BECKER Interface – voll parameterisiert: Firmenkopf, MWSt. und Rabattsätze, Größe der Dateien beliebig wählbar – 5 Zeilen für Firmenkopf je 30 Zeichen (erste Zeile erscheint auf der Rechnung in Breitschrift – 4 Mehrwertsteuer-Sätze; während der Rechnungsschreibung können also Artikel mit unterschiedlichem Mehrwertsteuer-Satz verrechnet werden – 10 Rabattsätze (Rabattsatz 1 vorbelegt mit 0%), bei der Rechnungsschreibung kann jedem Artikel ein Rabattsatz zugewiesen werden – maximal 1900 Artikel bei 50 Kunden oder 950 Kunden bei 100 Artikel (max. Artikel = (1000-Kunden)*2; max. Kunden = (2000-Artikel)/2) – manuelle Eingabe von Artikeln und/oder Kunde während der Rechnungsschreibung – d. h. es können mehr Artikel verrechnet weden als überhaupt in die Datei passen (bei Verzicht auf Lagerbuchführung) bzw. es können Rechnungen an Kunden geschrieben werden, die nicht erfaßt wurden – integrierte Lagerbuchführung mit Ausgabe einer Inventurliste – Rechnungsbeträge und Datum werden in der Kundendatei festgehalten – Druck von: Rechnung (mit Abbuchen aus Lager), Rechnung (ohne Abbuchen aus Lager), Lieferschein – deutsches detailliertes Handbuch mit Übungs- und Anwendungsteil – deutsche Bedienerführung innerhalb des Programms (z. B. „Artikel nicht vorhanden" anstelle „RECORD NOT PRESENT").

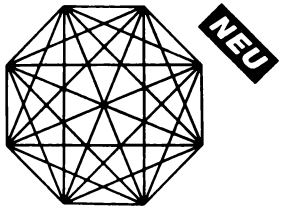
DM 148,–

UNI-TAB

Heute schon die Bundesliga-Tabelle von morgen kennen, das geht mit UNI-TAB. Alle Rechnereien, die man ohne dieses Programm nie machen würde, lassen sich in Sekundenschnelle durchführen. Wer will, kann mit simulierten Spielergebnissen den Weltmeister '86 vorausberechnen. Aber nicht nur Fußball-Ligen können tabellarisch erfaßt werden, fast alle Sportarten sind UNI-TAB-fähig. Gag am Rande: für viele Sportarten stehen die bekannten Piktogramme zur Verfügung.

UNI-TAB in Stichworten:
Menuesteuerung über die Funktionstasten mit leicht verständlichen Auswahlmöglichkeiten – Bedienerfreundlich (Mannschaften werden über Kennzahlen gesteuert) – Ligen mit 4 bis 20 Mannschaften können verwaltet werden (6 bis 38 Spieltage möglich) – unsinnige Ligen (z. B. 13 Mannschaften sollen 5 Spieltage absolvieren) sind ausgeschlossen – favorisierte Mannschaft kann während des Programmablaufs durch reverse Darstellung gekennzeichnet werden – Tabelle kann geändert werden (wichtig bei Spielanullierungen) – drei verschiedene Tabellenarten können abgespeichert und später eingelesen werden (die aktuelle Tabelle (unabhängig von der Vollständigkeit eines Spieltages), der komplette Spieltag (Vollständigkeit und Nummer des Spieltages werden automatisch errechnet), die simulierte Tabelle (der Anwender kann so selbst Schicksal spielen und seinen Tip später mit dem tatsächlichen Geschehen vergleichen)) – zwei verschiedene Arten der Saisonübersicht (die statistische Übersicht zeigt an, welchen Tabellenplatz das jeweilige Team bei welchem Punkte- und Torverhältnis an den einzelnen Spieltagen einnahm; die graphische Übersicht zeigt die Leistungskurve jeder Mannschaft) – alle Tabellen und Graphiken sind als Hardcopy auf einem Drucker darstellbar – bei Fehlbedienung (z. B. gewünschte Druckausgabe bei nicht eingeschaltetem Drucker) erscheinen leicht verständliche deutsche Fehlermeldungen.

DM 69,–



SUPERGRAFIK 64

Entdecken Sie die faszinierende Welt der Computergraphik mit SUPERGRAFIK 64, der starken Befehlserweiterung mit den vielseitigen Möglichkeiten. Durch die neue verbesserte Version jetzt noch leistungsstärker.

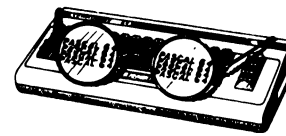
SUPERGRAFIK 64 in Stichworten:

2 unabhängige Graphikseiten (320 x 200 Punkte) – logische Verknüpfung der beiden Graphikseiten (AND, OR, EXOR) – 1 Standard Low-Graphik Seite (80 x 50 Punkte) – Normalfarben Graphik (300 x 200 Punkte) – Multicolor-Graphik (160 x 200 Punkte) – verdecktes Zeichnen (z. B. Text sichtbar, Graphikseite 2 wird erstellt) –Textfenster in der Graphik – 183 Befehle und Befehlskombinationen (1. Für jeden Befehl wählbare Zwischenmodi: Zeichnen, Löschen, Punktieren, Graphik-Cursor bewegen, Zeichnen mit/ohne Farbsetzung, Punkte zählen; 2. Durch einfache Befehle zu steuernde Graphikfiguren: Punkt, Linie, Linienschar, Linie vom Graphik-Cursor, Kreise, Kreisbögen, Ellipse, Ellipsenbögen, selbstdefinierbare Figuren, rotieren und vergrößern dieser Figuren, Rahmen, Feld, Text in Graphik; 3. Weitere Graphikbefehle: Graphikseiten- und Moduswechsel, Graphik löschen, Graphik invertieren, Scrolling von Text und Graphik, Wählen der Rahmen-, Hintergrund-, Zeichen- oder Punktfarbe) – Speichern, Laden von Graphik (auch verdeckt) – Kopieren des Textbildschirms in die Graphikseite – Hardcopies für EPSON, Seikosha GP100VC, Farb(!)drucker Seikosha GP700 und andere mit DATA BECKER Interface – 16! Sprites gleichzeitig auf dem Bildschirm – alle Sprite-Eigenschaften veränderbar – Positionieren und Bewegen (!) von 16 Sprites gleichzeitig und unabhängig voneinander, während das übrige Programm weiterläuft (IRQ) – Sprite-Kollisionsüberprüfung, Joystickunterstützung – automatische Unterbrechung des BASIC-Programms bei Kollisionen (Interrupt), Sprung in Unterbrechungsroutine, dann Weiterführung des Hauptprogramms – komfortable Soundprogrammierung mit Verstellung aller möglichen Sound-Parameter (Lautstärke, Klang, Filter, Tonhöhe, Tonlänge), ebenfalls unabhängig vom übrigen Programmlauf – zahlreichen Programmiertools (MERGE, RENUMBER usw.) – umfangreiche Anleitung – Diskettenprogramm.

DM 99,–

PASCAL 64

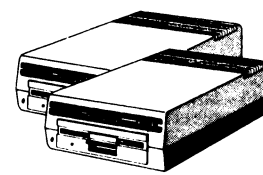
Beim Wort „Compiler" fällt dem Eingeweihten sicher der Begriff „Geschwindigkeit" ein. Ein PASCAL-Compiler sollte jedoch weitere Assoziationen wecken. Strukturiertes Programmieren heißt das Zauberwort. PASCAL wurde eigens zu didaktischen Zwecken entwickelt und erfüllt diese Aufgabe auch heute noch. Der PASCAL 64 Compiler bringt diese phantastische Programmiersprache auf den 64er.
Gerade die neue, verbesserte Version unterstützt die Möglichkeiten des C-64 in jeder Hinsicht und macht leistungsfähige Programme möglich.



PASCAL 64 in Stichworten:

besitzt einen sehr umfangreichen Befehlsvorrat – erlaubt Interruptprogrammierung und bietet Schnittstellen zu Monitor und Assembler – erzeugt sehr schnelle Programme in reinem Maschinencode – unterstützt relative Dateiverwaltung; Graphik und Sound – bietet die Datentypen REAL, INTEGER, CHAR und BOOLEAN sowie Aufzähltypen und POINTER, die zu Datenstrukturen RECORD, SET, ARRAY und PACKED ARRAY kombiniert werden können – erlaubt vorzeitigen Abschluß von Prozeduren mit EXIT, uneingeschränkte Rekursionen und komfortable Verarbeitung von Teilfeldern (Strings) – ist ein ausgereiftes, deutsches Produkt und wird mit ausführlichem Handbuch geliefert.

DM 99,–



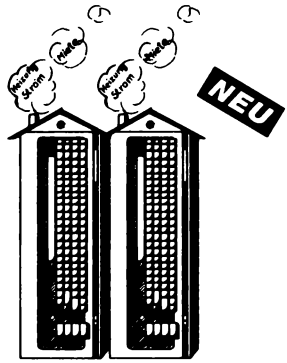
DISKOMAT

Der Umgang mit Diskettenlaufwerken ist für viele noch immer mit Geheimnissen belastet. Andere stören sich an den wenig komfortablen Diskettenbefehlen des BASIC V2. DISKOMAT bringt Abhilfe; alle Diskettenbefehle des BASIC 4.0 stehen zur Verfügung. Außerdem können mit dem Programm SUPERTWIN zwei 1541-Laufwerke wie ein Doppellaufwerk verwaltet werden. Für Benutzer, die sich die Fähigkeiten der Floppy 1541 ganz erschließen wollen, steht der DISK-MONITOR bereit; er macht es endlich möglich, den direkten Zugriff auf einzelne Blocks einfach und bequem vorzunehmen.

DISKOMAT in Stichworten:

Diskettenprogramm – DISK BASIC unterstützt Diskettenbefehle des BASIC 4.0 (CONCAT, HEADER, APPEND, RENAME, OPEN, COLLECT, DSAVE, SCRATCH, DCLOSE, BACKUP, DLOAD, DIRECTORY, RECORD, COPY, CATALOG, DS & DS$) – SUPER TWIN behandelt 2 Laufwerke 1541 wie ein Doppellaufwerk – DISK-MONITOR ermöglicht direkte Analyse und Manipulation von Disketten (direktes Lesen und Schreiben einzelner Blöcke, ändern von Blöcken mittels Bildschirm-Editor, Anzeige des Diskettenstatus, direktes Absenden von Disketten-Befehlen) – ausführliches deutsches Handbuch beschreibt jeden einzelnen der 3 Programmteile.

DM 99,–

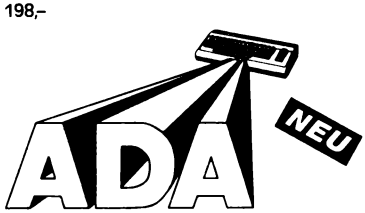


HAUSVERWALTUNG

Jetzt können alle Hausbesitzer aufatmen: das Programm HAUSVERWALTUNG bietet Ihnen eine sehr komfortable Verwaltung der Mietwohnungen mit dem COMMODORE 64.
Alles, was Sie dazu brauchen, ist ein COMMODORE 64, ein Diskettenlaufwerk 1541, ein anschlußfähiger Drucker und das obengenannte Programm HAUSVERWALTUNG. Die nachfolgenden und viele weitere leistungsfähige Features ermöglichen eine äußerst rationelle Verwaltung Ihrer Mietwohnungen.

HAUSVERWALTUNG in Stichworten:
Dikettenprogramm – Verwaltung von 50 Einheiten pro Objekt möglich – Stammdatenverwaltung für Häuser und Mieter – Verbuchen der Miete, Nebenkosten und Garagenmieten – Mietkontoanzeige – Haus- und Mieteraufstellung – Mahnungen – Verbuchen der anfallenden Kosten – Kostengegenüberstellung – Jahresendabrechnung mit automatischem Jahresübertrag – umfangreiches deutsches Handbuch.

DM 198,–

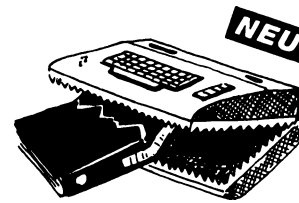


TRAININGSKURS zu ADA

Diese Programmiersprache der Zukunft, die das Pentagon in Auftrag gegeben hat, wird jetzt durch DATA BECKER auch dem C-64 Anwender zugänglich gemacht durch den TRAININGSKURS zu ADA, der eine sehr gute Einführung in diese Supersprache bietet. Der dazu gelieferte Compiler liefert ein umfangreiches Subset der Sprache.

ADA in Stichworten:
blockstrukturierte Programme – modularer Aufbau der Programme – ermöglicht die Behandlung von Ausnahmezuständen – Fehlerüberprüfung beim Übersetzen und zur Laufzeit – ermöglicht das einfache Einbinden von Maschinenprogrammen – sehr leichtes Arbeiten mit Programmbibliotheken – Programmdiskette enthält Editor, Übersetzer, Assembler und Disassembler – umfangreiches deutsches Handbuch.

DM 198,–



DATAMAT

Daten verwalten kann ein schier endloses Hantieren mit Karteikästen und Aktenordnern bedeuten; kann aber auch C-64 plus DATAMAT heißen. Dann wird Suchen und Sortieren zum Spaß. Der DATAMAT bietet in seiner neuen Version einiges, was in dieser Preisklasse bisher unvorstellbar schien. Nicht nur Geschwindigkeit und Bedienungsfreundlichkeit wurden weiter verbessert, auch die Anpassung an die meisten Drucker ist inzwischen machbar.

DATAMAT in Stichworten:
menuegesteuertes Diskettenprogramm, dadurch extrem einfach zu bedienen – für jede Art von Daten – völlig frei gestaltbare Eingabemaske – 50 Felder pro Datensatz – 253 Zeichen pro Datensatz – bis zu 2000 Datensätze pro Datei je nach Umfang – Schnittstelle zu TEXTOMAT – läuft mit 1 oder 2 Floppies – völlig in Maschinensprache – extrem schnell – deutscher Zeichensatz auch auf COMMODORE-Druckern – fast jeder Drucker anschließbar – ausdrucken über RS 232 – duplizieren der Datendiskette – verbesserte Benutzerführung – Hauptprogramm komplett im Speicher (kein Diskettenwechsel mehr) – integrierte Minitextverarbeitung – deutsches Handbuch mit Übungslektionen
Sie können:
jeden Datensatz in 2 – 3 Sekunden suchen – nach beliebigen Feldern selektieren – nach allen Feldern gleichzeitig sortieren – Listen in völlig freiem Format drucken – Etiketten drucken.

DM 99,–

ZAHLUNGSVERKEHR

Umfangreicher Zahlungsverkehr kann zur Plage werden. Das Software-Paket ZAHLUNGSVERKEHR übernimmt den größten Teil dieser Arbeit. Außer den notwendigen Fähigkeiten für das Ausfüllen und Auflisten von Überweisungen und Schecks ist der ZAHLUNGSVERKEHR in der Lage, Sammellisten, Einzugslisten etc. selbständig zusammenzustellen.

ZAHLUNGSVERKEHR in Stichworten:
Diskettenprogramm – max. 100 Zahlungsempfänger pro Diskette – drei definierbare Absenderbanken – 25 Zahlungsdateien – 14 frei definierbare Formulare – Kontrolldruck bei Belegeingabe möglich – Eingabe von Rechnungsdaten oder eines Verwendungszwecks – Ausdruck einer Sammel-Überweisungsliste – Korrekturmöglichkeit der einzelnen Zahlungsdateien – arbeitet mit einer oder zwei Floppies – umfangreiches deutsches Handbuch.

DM 148,–

DAS STEHT DRIN:

Das große FLOPPYBUCH ist ein Standardwerk zur Disketten-Programmierung mit COMMODORE Computern für Anfänger, Fortgeschrittene und Profis. Ein Buch, das Ihnen ausführlich und verständlich die Arbeit mit der Floppy VC-1541 erklärt.

Aus dem Inhalt:

- Floppy-Systembefehle
- Speichern von Programmen
- Sequentielle und relative Datenspeicherung
- Fehlermeldungen und ihre Ursachen
- Direktzugriff
- DOS-Listing der VC-1541
- BASIC-Erweiterungen und Programme
- Overlaytechnik
- Diskmonitor
- IEC-Bus und serieller Bus

UND GESCHRIEBEN HABEN DIESES BUCH:

Lothar Englisch ist Systemprogrammierer in der DATA BECKER Entwicklungsabteilung. Vor allem aber ist er der Autor vieler beliebter DATA BECKER Bücher (Das Maschinensprachebuch, 64 Intern . . .).
Norbert Szczepanowski, EDV-Kaufmann, ist ebenfalls Bestsellerautor bei DATA BECKER (Trainingsbuch zu SIMON'S BASIC, 64 für Einsteiger). Natürlich ist er begeisterter Programmierer.

ISBN 3-89011-005-3

This was brought to you

from the archives of

http://retro-commodore.eu